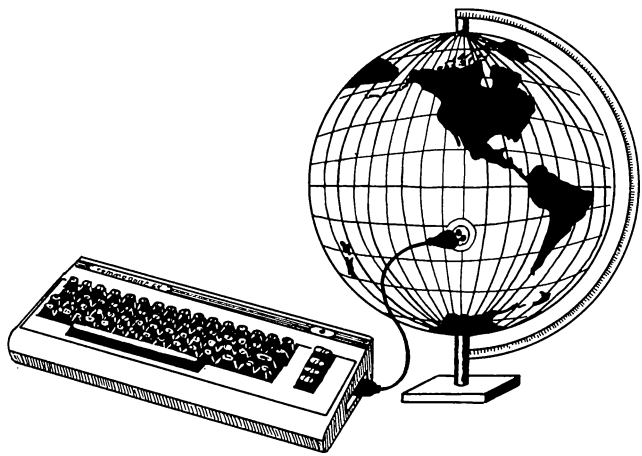
Brückmann

DER COMMODORE 64 UND DER REST DER WELT

EIN DATA BECKER BUCH

Brückmann

DER COMMODORE 64 UND DER REST DER WELT



EIN DATA BECKER BUCH

ISBN 3-89011-015-0

Copyright (C) 1984 DATA BECKER GmbH
Merowingerstr. 30
4000 Düsseldorf

Alle Rechte vorbehalten. Kein Teil dieses Buches darf in irgendeiner Form (Druck, Fotokopie oder einem anderen Verfahren) ohne schriftliche Genehmigung der DATA BECKER GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Wichtiger Hinweis!

Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technische Angaben und Programme in diesem Buch wurden von den Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler sind die Autoren jederzeit dankbar.

Vorwort

Wer an Computer denkt, der denkt auch gleich an Programmieren, an Spiele und kommerziellen Einsatz. Außer Acht gelassen wird häufig, daß sich gerade die modernen Mikrocomputer durch ihr hervorragendes Preis-Leistungsverhältnis und Vielseitigkeit sehr gut für technische Anwendungen eignen. Dies gilt sowohl für den Hobbybereich als auch für den professionellen Einsatz. Literatur hierzu ist allerdings recht dünn gesät. Diese Lücke wollen wir mit diesem Buch schließen. Geschrieben wurde es von Rolf Brückmann, einem engagierten Techniker, für den Computer Hobby und Beruf zur gleichen Zeit ist. Im Jahre 1980 stieg er auf bemerkenswerte Art und Weise in die Welt des Computers ein. Er kaufte sich nicht einfach ein Gerät - wie dies heute üblich ist - sondern baute sich selber eins, einen Computer auf 8080-Basis. Auch alle in diesem Buch vorhandenen Schaltungen hat Rolf Brückmann komplett selbst entwickelt. Er kennt nicht nur den COMMODORE 64 wie seine eigene Westentasche, sondern hat auch fundiertes Wissen auf allen angrenzenden Gebieten. Dieses Wissen gibt er mit diesem Buch, das sicher nicht sein letztes ist, gerne an Sie weiter. Ich darf Ihnen viel Spaß beim Lesen dieses Buches und viel Erfolg mit Ihren eigenen Anwendungen wünschen.

Dr. Achim Becker

Inhaltsverzeichnis

1	Einleitung	1
2.0	Wie Computer rechnen	4
2.1	PEEK, POKE & CO.	10
3.0	Anschlußmöglichkeiten am C64	15
3.1	USER - Port	16
3.2	EXPANSION - Port	18
4.0	Die CIAs 6526	20
4.1	LEDs am USER - Port	28
4.2.0	Schalten mit Transistor	32
4.2.1	Schalten mit Relais	33
4.2.2	Ansteuerung von Optokopplern	38
4.2.3	TRIAC-Schaltstufe mit Optokoppler	39
4.2.4	Der 64 als Lichtorgel	41
4.3	Eingabe von Daten in den USER - PORT	44
4.4	Die Lichtschranke	46
4.4.1	Zählen von Impulsen mit Lichtschranke	48
4.4.2	Stopuhr mit Lichtschranke	51
4.5	Verwendung der Timer	66
4.6	Klatschschalter	71
4.7	Tonerzeugung mit dem Cassetten-Port	74
4.8.1	Motorsteuerung Gleichstrommotoren	78
4.8.2	Motorsteuerung Schrittschaltmotoren	81
5.0	Anschluß weiterer Peripherie-ICs	87
5.1	Anschluß weiterer 6526	88
5.2	Die VIA 6522	90
5.3	Der Parallelschnittstellenbaustein 8255	103
5.4	Andere Peripherie-ICs	107

6.0 D/A-Wandlung - was ist das? 108
6.1 Die Methoden der D/A-Wandlung 109
6.2 Der ZN 426, ein universeller Wandler 111
6.2.1 Programmierbare Stromversorgung 113
6.2.2 Motorregler mit ZN 426 118

7.0 A/D-Wandlung - der umgekehrte Weg 120
7.1 Methoden der A/D-Wandlung 120
7.2 Der eingebaute A/D-Wandler 126
7.3 Der ZN 427 127
7.4 Der CA 3162 135
7.4.1 Spannungsmessung 143
7.4.2 Die Messung von Gleichströmen 145
7.4.3 Temparaturmessung 146

8.0 Grundsätzliches zur Interrupttechnik 148
8.1 Es laufen zwei Dinge gleichzeitig 153

9.0 Komplette Schaltungen zum Selberbauen 159
9.1 EPROM-Programmiergerät 160
9.2 EPROM-Karte für den C64 172
9.3 Ein Frequenzzähler bis 30 MHz 176
9.4 Auto - DVM 187
9.5 Logic-Analyser 197
9.6 Der C64 lernt sprechen 206

Anhang 219

1.0 Einleitung

Der 64 und der Rest der Welt - ein vielleicht etwas verblüffender Titel für ein Buch.

Gemeint ist damit aber nicht die Verwendung des C64 in anderen Ländern der Erde oder ein Vergleich des Gerätes mit seinen Konkurenten im In- und Ausland.
Vielmehr soll in diesem Buch der Rest der Welt aus der Sicht des Computers beschrieben werden.
Dieser 'Rest' umfasst das weite Gebiet der Digital- und Analogtechnik.
Beginnend mit den Grundlagen der Computerei, den Zahlensystemen, werden Sie in diesem Buch schrittweise von einfachen Experimenten bis zu komplexen Meß- und Steuerschaltungen und der entsprechenden Programmierung geführt.
Dabei werden Sie sowohl Meßgeräte wie Frequenzzähler oder Digitalvoltmeter als auch Schaltungen mit solch experimentellem Charakter wie Sprachein-ausgabe oder Vorschläge zur Realisation eines computerisierten Klatschschalters kennenlernen.

Allerdings werden an Sie, den Leser, auch Anforderungen gestellt.
So sollten Sie in der Lage sein, Schaltpläne zu lesen und sich in der Bezeichnung von handelsüblichen Bauteilen wie Widerständen oder Kondensatoren auskennen.
Sie sollten wissen, was ein Inverter oder ein Nand-Gatter ist und das TTL-ICs eine Beriebsspannung von 5 Volt benötigen.
Wenn Sie wissen, daß Ihr C64 nicht kaputt ist, wenn seine stereotype Antwort auf Ihre Eingaben 'SYNTAX ERROR' ist, dann haben Sie zumindest für den Anfang ausreichende Basic-Programmierkenntnisse, da die Programme in den ersten

Kapiteln alle ausführlich besprochen werden. Für die späteren Kapitel sind allerdings weitergehende Kenntnisse dann von Vorteil, wenn Sie die Programme nicht nur abtippen, sondern auch wissen wollen, warum das Programm funktioniert. In einigen Kapiteln konnte eine Programmierung in Maschinensprache aus Geschwindigkeitsgründen nicht vermieden werden. Wenn Sie bis jetzt die Maschinensprache gemieden haben, weil Sie der Ansicht waren, Basic sei für Ihre Anwendungen schnell genug, dann ist dies Buch vielleicht ein guter Anreiz, es doch einmal mit dieser sehr schnellen und effektiven Programmierung zu versuchen. Sie werden bald feststellen, daß gerade im Zusammenspiel von Hard- und Software die Maschinensprache oft unabdingbar ist. Zusätzlich sind aber alle Maschinenspracheroutinen auch als Basic-Listing mit DATA-Zeilen abgedruckt.
Um die Schaltungen aufzubauen, benötigen Sie natürlich eine Grundausstattung an Werkzeug. Ein Lötkolben mit feiner Spitze (30 Watt), einige Zangen und eine Pinzette sowie ein Vielfachmeßgerät sind für den Anfang ausreichend.
Die Stromversorgung der Zusatzschaltungen kann im Normalfall dem C64 entnommem werden. Trotzdem ist ein zusätzliches kurzschlußfestes Netzgerät mit den Ausgangsspannungen +5 Volt und +12 Volt von Vorteil, da die Stromaufnahme der aufgebauten Schaltungen dann zuerst einmal am Netzteil geprüft werden kann.

Alle (!) Schaltungen und Programme wurden ausgiebig getestet und für nachbausicher gefunden.
Wenn also eine Schaltung oder das dazugehörige Programm einmal nicht auf Anhieb klappen, dann schimpfen Sie bitte nicht gleich auf dies Buch oder den Autor.
Überprüfen Sie in aller Ruhe Ihren Aufbau und das Programm und versuchen Sie die Fehlerquelle so genau wie möglich zu lokalisieren. Dann sollte Ihnen mit Hilfe der Funktions-

beschreibung die Beseitigung des Fehlers möglich sein.
Wichtiger als teure Meßgeräte sind bei der Fehlersuche geplantes und konsequentes Vorgehen und etwas Phantasie!

Zum Abschluß soll ein praktischer Tip zum Aufbau folgen:
Wenn Sie die Schaltungen auf einer Platine (Lochraster oder geätzt) aufbauen, dann sollten Sie nach Möglichkeit alle ICs auf Sockel setzen. Die Sockel kosten nur ein paar Pfennige und ersparen eventuelles späteres umständliches Auslöten von Bauteilen.
Für den schnellen Aufbau kann nur die Anschaffung eines Versuchssteckbretts empfohlen werden. Diese Steckbretter sich im Elektronik-Fachhandel zu bekommen und ermöglichen einen sehr schnellen Aufbau kleinerer Schaltungen. Änderungen sind blitzartig vorgenommen und die Bauteile lassen sich immer wieder verwenden.

Ich wünsche Ihnen und Ihrem C64 eine gute Kommunikation mit dem Rest der Welt.

2.0 Wie Computer rechnen

Wenn Sie jetzt schon Ihren Lötkolben eingeschaltet haben und auf die ersten Bauvorschläge warten, dann müssen wir Sie an dieser Stelle noch um ein klein wenig Geduld bitten.
Zuerst wollen wir uns einen kurzen Moment mit der trockenen Theorie befassen.
Das Thema sind die verschiedenen Zahlensysteme, also Binär-, Hexadezimal- und Dezimalzahlen. Sie mögen jetzt denken, die Dezimalzahlen seien für uns noch immer gut genug. Schließlich haben wir ja zehn Finger und dieses Zahlensystem ist uns mit viel Mühe in der Schule nahegebracht worden.
Aber die Binär- und Hexadezimalzahlen sind gerade im Zusammenhang mit der Programmierung von Peripheriegeräten besonders wichtig. Ohne ausreichende Kenntnis dieser Zahlensysteme wird das Verstehen der Funktion vieler Schaltungen aus diesem Buch erschwert.
Aber betrachten wir doch einmal, wie unser so vertrautes 10er System funktioniert.
Wenn wir eine Zahl mit den Fingern zählen, dann beginnen wir mit dem Daumen und zählen so lange, bis wir keinen freien Finger mehr haben. Dieser Moment ist bei der Zahl 10 erreicht. Jetzt merken wir uns diesen 'Überlauf', also eins im Sinn, und beginnen wieder mit dem Daumen.
Diese Überläufe sehen wir uns an einem Beispiel noch einmal etwas genauer an. Stellen wir uns einmal einen Betrag von 247 DM in Markstücken vor.
Wir gehen jetzt so vor, daß immer, wenn wir 10 Markstücke gezählt haben, wir diese aufeinander stapeln.
Nach einiger Zeit werden wir vor uns eine Menge Münzstapel und 7 einzelne Markstücke liegen haben. Die 7 einzelnen Münzen lassen wir jetzt liegen und wenden uns den Stapeln a 10 DM zu. Auch diese Stapel können wir wieder mit den Fingern zählen, wobei wir bei jedem 'Überlauf' die gezählten

Münzen zu einem hohen Stapel zusammenstellen. Wenn wir diese Arbeit richtig hinter uns gebracht haben, dann sollten jetzt 2 Stapel mit 100 Markstücken und 4 kleine Stapel mit 10 Markstücken vor uns liegen. Zusätzlich haben wir noch die 7 einzelnen Markstücke.
Wenn wir uns jetzt die Zahl 247 anschauen, erkennt man sehr deutlich, wie die Stellung einer Ziffer innerhalb der Zahl die Wertigkeit bestimmt.

```
                2 1 0
Anzahl der Stapel 2 4 7
                : : :
                : : : .... mal 1   = 10^0
                : : ...... mal 10  = 10^1
                : ........ mal 100 = 10^2
```

Die Wertigkeiten der einzelnen Stellen errechnet sich also aus der Stelle als Exponent und der Grundzahl des Zahlensystems als Basis.

Ihr Computer hat als Grundzahl die 2. Er rechnet nicht mit 10 Fingern, er kennt nur die elektrischen Zustände Spannung da und Spannung nicht da.

Die Frage ist, wie man mit nur zwei Zuständen zählen kann. Wir können das einfach nachvollziehen, indem wir uns vorstellen, wir hätten nur unsere beiden Daumen zum Zählen. Bei der dritten Mark hätten wir bereits einen Überlauf. Diesem Überlauf begegnen wir wieder in der selben Weise wie zuvor. D.h. wir stapeln die zwei Münzen aufeinander und beginnen von vorn.
Wieder nach einiger Zeit bekommen wir so 123 Stapel mit zwei Markstücken und eine einelne Münze. Diese einzelne Münze legen wir jetzt beiseite. Die kleinen Stapel werden wieder

gezählt und wir bekommen 61 Stapel mit 4 Geldstücken. Auch diesmal legen wir den Reststapel an die Seite.
Wenn wir so weiterzählen, dann lautet das nächste Ergebnis 30 Stapel mit 8 Mark und ein Reststapel mit 4 Mark, den wir zu den anderen Reststapeln legen. Die 30 verbliebenen Stapel teilen sich auf in 15 Stapel a 16 Mark. In diesem Fall haben wir keinen Rest. Aus diesen 15 Stapeln werden 7 a 32 DM und ein Rest, daraus werden 3 a 64 DM und ein Reststapel und daraus zu guterletzt ein Stapel mit 128 DM und ein Reststapel mit 64 DM.
Wenn wir jetzt diese Stapel wie die Orgelpfeifen vor uns aufbauen, wobei der höchste Stapel nach links, der kleinste nach rechts kommt, dann sieht das ganze etwa so aus:

```
                7 6 5 4 3 2 1 0
Anz. der Stapel 1 1 1 1 1 0 1 1
                : : : : : : : :
                : : : : : : : : ..... mal   1 = 2^0
                : : : : : : : ..... mal   2 = 2^1
                : : : : : : ..... mal   4 = 2^2
                : : : : : ..... mal   8 = 2^3
                : : : : ..... mal  16 = 2^4
                : : : ..... mal  32 = 2^5
                : : ..... mal  64 = 2^6
                : ..... mal 128 = 2^7
```

So, jetzt haben wir ganz nebenbei eine Dezimalzahl in eine Binärzahl umgewandelt. Die Dezimalzahl 247 ist also gleich mit der Binärzahl 11111011.

Umgekehrt können wir jetzt jede achtstellige Binärzahl an Hand des obigen Schemas in eine Dezimalzahl wandeln. Das wollen wir auch gleich einmal machen, um die höchste achtstellige Binärzahl zu bestimmen.

Da in einem zweiwertigen Zahlensystem nur die Ziffern 0 und 1 möglich sind, ist die größte mögliche Zahl also 11111111. Wenn wir diese Zahl nach dem obigen Schema aufschlüsseln, dann erhalten wir:

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

Die kleinste Zahl würde aus 8 Nullen bestehen, wäre also auch Null.
Nun ist dieses Beispiel natürlich nicht zufällig entstanden. Es hat schon seinen Grund, daß die entstandene Binärzahl gerade acht Stellen groß war.
Der im 64 eingesetzte Microprozessor rechnet mit diesen 8-stelligen Binärzahlen, die auch Dualzahlen genannt werden. Er ist ein sogenannter 8-Bit-Prozessor. Die Bezeichnung Bit steht für eine dieser acht Stellen. Ein Bit ist die kleinste vorstellbare Möglichkeit um unterschiedliche Information zu speichern und darzustellen. Die Information kann entweder ein Ja oder ein Nein sein.
Dieses Ja oder Nein haben wir gerade benutzt, um darzustellen, ob wir einen Münzstapel mit der entsprechenden Wertigkeit hatten, oder nicht. Dazu haben wir für jeden vorhandenen Stapel eine 1 geschrieben, für jeden nicht vorhandenen Stapel der entsprechenden Wertigkeit eine 0.
In der Digitaltechnik ist auch noch eine andere Bezeichnung üblich, HIGH und LOW.
Diese Bezeichnungen signalisieren den vorhandenen Spannungspegel. Wenn ein Bit gesetzt oder anders ausgedrückt auf 1 ist, kann man eine Spannung messen (in Ihrem 64 sind das zwischen +2.8 und +5 Volt). In diesem Fall spricht man von einem High-Signal. Ist ein Bit dagegen nicht gesetzt oder auf logisch 0, dann kann man keine Spannung messen, man spricht von einem Low-Signal (als Low gilt im 64 ein Spannungspegel von < +0.8 Volt).

Da der Prozessor also mit 8-Bit-Werten rechnet (8 Bit werden normalerweise Byte genannt), ist der Zahlenbereich für den Prozessor auf die Werte zwischen 0 und 255 festgelegt.

Bleibt noch die Erklärung des Begriffes Hexadezimalzahlen.
Hexadezimal ist genau wie dezimal, nur mit dem Unterschied, das Sie 16 Finger haben, mit denen Sie Zählen können.
Da wir aber nur für 10 Finger Bezeichnungen oder Ziffern haben (0 - 9), müssen wir uns eines Tricks bedienen. Wir verwenden einfach die ersten 6 Buchstaben des Alphabets als Ziffern.
In Hex (der gebräuchlichen Abkürzung für Hexadezimal) zählt man 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
Dabei gilt A = 10, B = 11 ..... F = 15. Dezimal 16 ist in Hex dargestellt 10. Genau wie bei den beiden anderen Zahlensystemen haben wir einen Überlauf den wir uns in einer entsprechend höher bewerteten Stelle merken.
Ganz nett, aber was ergibt das für einen Sinn?
Nun, wenn wir uns einmal eine einstellige Hex-Zahl ansehen, dann stellen wir fest, das wir damit 16 verschiedene Zustände kennzeichnen können. Wenn wir eine beliebige Hexzahl in die entsprechende Binärzahl wandeln, dann ist diese Binärzahl maximal 4 Stellen lang. Teilen wir jetzt den 8-Bit-Datenbus des Prozessors in zwei 4-Bit-Hälften (eine solche 4-Bit-Hälfte wird übrigens Nibble genannt), dann können wir jede Binärzahl in einer zweistelligen Hexzahl darstellen.
Der große Vorteil : 9E ist sicher leichter zu merken als 10011110.
Allerdings ist die Bezeichnung Hexadezimal, obwohl gebräuchlich, nicht exakt. Es müßte richtig Sedezimal heißen. Wir werden jedoch weiterhin den zwar falschen aber vertrauten Begriff Hexadezimal weiter benutzen.
Bei so vielen Zahlensystemen ergibt sich jetzt eine

Schwierigkeit. Wie unterscheidet man, welchem System eine Zahl zugeordnet ist? 10 kann sowohl dezimal 10 sein als binär 10, was dann dezimal 2 entpricht. 10 kann aber auch im Hexsystem gemeint sein, diese Zahl wäre dann gleich 16 dezimal.
Zur sicheren Unterscheidung der verschiedenen Systeme hat man sich auf einen Standard geeinigt, den wir im folgenden ebenfals verwenden wollen. Danach gilt :

%10	Binärzahl
$10	Hexzahl
10	Dezimalzahl

Das Prozentzeichen weist eine Zahl damit als Binärzahl aus, das Dollar-Zeichen eine Zahl als Hexzahl. Nur wenn kein Zeichen vor der Zahl steht, ist es eine Dezimalzahl.
Leider gilt diese Vereinbarung nur so lange, wie Sie mit den 65xx-Prozessoren arbeiten. Sollten Sie dagegen auch mit 8080 oder Z80 (in der CP/M-Cartridge) arbeiten, dann werden Hexzahlen durch ein vorangestelltes H gekennzeichet.
In diesem Buch werden wir aber bei der 65xx-Vereinbarung bleiben.
Zum Abschluß unseres kleinen Ausflugs in die Welt der Zahlensysteme sei hier kurz und nur der Vorständigkeit halber das Oktalsystem erwähnt. Dieses Zahlensystem basiert auf der Zahl 8 und hat seinen Ursprung in den ganz frühen Tagen der Microcomputerei.
Im Oktalsystem sind nur die Ziffern 0 bis 7 zugelassen, es lassen sich also maximal 8 verschiedene Zustände erreichen. Dafür sind nur drei Bits erforderlich.

2.1 PEEK, POKE & CO.

Dieses Buch ist der Hardware sowie dem Anschluß des C64 an Zusatzschaltungen gewwidmet. Darum sollen die folgenden Seiten keinen BASIC-Schnellkurs darstellen. Wir wollen aber an dieser Stelle einige, für unsere Zwecke besonders wichtige Befehle betrachten.
Der erste Befehl, den wir uns anschauen wollen, ist der POKE-Befehl.
Die Syntax des POKE-Befehls lautet :

```
POKE adresse,wert
```

Mit diesem Befehl können wir die Inhalte einzelner Speicherzellen verändern.
Die Begiffe 'adresse' und 'wert' stehen für einen Zahlenwert oder eine Variable. 'adresse' steht für die physikalische Adresse im Speicherbereich des Prozessors. Um eine Adresse im Speicher anzusprechen, stehen dem Prozessor 16 Leitungen zur Verfügung. Jede dieser Leitungen kann entweder HIGH oder LOW sein, stellt also ein Bit dar. Mit 16 Bits ergibt sich ein Zahlenbereich von 0 bis 65535. Der Prozessor kann somit 65536 verschiedene Speicherzellen ansprechen.
Jede dieser Speicherzellen ist 8 Bit breit und kann, wie wir wissen, 256 verschiedene Zustände einnehmen. Damit ist auch schon die Größe von 'wert' festgelegt. 'wert' kann jede Zahl zwischen 0 und 255 sein.
Mit dem Befehl POKE 49152 , 255 wird in die Speicherzelle 49152, entsprechend hexadezimal $C000 der Wert 255 geschrieben. Der zuvor in der Speicherzelle enthaltene Wert ist überschrieben und verloren.

Um den Inhalt einer Speicherzelle zu bestimmen, verwenden wir den Befehl PEEK.

```
wert = PEEK (adresse)
```

In diesem Fall steht der Ausdruck 'wert' natürlich für eine Variable, in der der Inhalt der durch 'adresse' angegebenen Speicherzelle zur Verfügung steht.
Bei jedem Versuch, eine Zahl größer 65535 oder kleiner 0 als Adresse anzugeben reagiert der Computer mit einer Fehlermeldung, da diese Werte ja ausserhalb der durch die 16 Adressbits vorgegebenen Grenzen liegt.

Die nächsten Befehle, die wir kurz betrachten wollen, sind die sogenannten logischen Verknüpfungsbefehle AND, OR und NOT.

Diese Befehle geben uns die Möglichkeit in die Hand, einzelne Bit innerhalb eines Byte gezielt zu beeinflussen.
In einem kleinen Beispiel wollen wir uns die Wirkung dieser Befehle verdeutlichen.
Dazu denken wir uns 8 Relais, mit denen wir in unserer Wohnung das Licht, die Kaffeemaschine, das Heißwassergerät u.s.w. ein und ausschalten können. Diese 8 Relais seien einmal auf der Speicheradresse 56577 zu erreichen. Dabei steuern wir jedes Relais mit einem Bit des Datenbusses so, das ein High-Bit das Relais anziehen läßt, ein auf Low gesetztes Bit das Relais abfallen läßt.
Um unsere Relais zu bezeichnen, nummerieren wir sie von 0 bis 7 durch. Diese Bezeichnungen sind damit gleich mit den Bezeichnungen der Bits des Datenbusses.

Datenbit	7	6	5	4	3	2	1	0
Relais	7	6	5	4	3	2	1	0

Wollen wir Relais 4 ansteuern, dann bekommen wir das folgende Bitmuster:

Datenbit	7	6	5	4	3	2	1	0
	0	0	0	1	0	0	0	0

Dieses Bitmuster müssen wir jetzt in eine Dezimalzahl wandeln. Da nur ein Bit gesetzt ist, ergibt sich die Dezimalzahl aus der Stelle des Bits als Exponent zur Basis 2. Das Ergebnis lautet also 2↑4 = 16.
Mit POKE 56577,16 haben wir das Relais mit der Nummer 4 eingeschaltet.
Sollen die Relais 6 und 2 eingeschaltet werden, dann lautet das Ergebnis POKE 56577,2↑6+2↑2 oder POKE 56577,64+4.
Aber kommen wir jetzt zu den logischen Verknüpfungen. Mit unserem vorigen Beispiel haben wir die Relais 6 und 2 eingeschaltet. Jetzt wollen wir auch noch das Relais 7 anschalten. Dazu eignet sich die ODER-Verknüpfung die im BASIC-Interpreter im OR-Befehl realisiert ist.
Im einfachsten Fall besteht eine ODER-Verknüpfung aus zwei parallel geschalteten Lichtschaltern. Wenn der eine ODER der andere Schalter eingeschaltet ist, dann leuchtet die angeschlossene Lampe. Genau so verhält sich der OR-Befehl in Ihrem Computer. Schauen wir uns dazu noch einmal die beiden Bitmuster an.

Datenbit	7	6	5	4	3	2	1	0		
	0	1	0	0	0	1	0	0	=	64
ODER	1	0	0	0	0	0	0	0	=	128
	1	1	0	0	0	1	0	0	=	196

Wenn in dem einen ODER in dem anderen Operanden ein Bit

gesetzt ist, dann ist das entsprechende Bit auch im Ergebnis gesetzt.
Aufmerksame Leser werden jetzt vielleicht sagen, daß dieses Ergebnis auch durch eine einfache Addition zu erreichen gewesen wäre. Das ist in diesem Fall richtig, wichtig ist aber, das sich unser Ergebnis nicht ändert, wenn wir es noch einmal mit dem Wert 128 ODER-verknüpfen. Würden wir ein zweites mal 128 addieren, wäre das Ergebnis 324, ein Wert, der als Argument in einer POKE-Anweisung eine Fehlermeldung (ILLEGAL QUANTITY ERROR) produziert. Bei kleineren Werten als 128 führt die Addition dazu, daß ganz andere als die gewünschten Relais schalten.
In BASIC lautet die Programmzeile also:

```
POKE 56577,PEEK (56577) OR 128
```

Mit der PEEK-Anweisung haben wir uns den aktuellen Zustand der Relais besorgt, dann mit der OR-Anweisung das Bit 7 gesetzt und mit der POKE-Anweisung den neuen Wert zurückgeschrieben.
Damit haben wir die Möglichkeit, alle Relais unabhängig von den anderen einzuschalten.
Um die angeschlossenen Geräte aber genau so unabhängig wieder auszuschalten, eignet sich dieser Befehl nicht. Für diesen Zweck müssen wir die UND-Verknüpfung zur Hilfe nehmen.
Um bei den Lichtschaltern von vorhin zu bleiben, können wir uns eine UND-Verknüpfung als zwei hintereinander geschaltete Schalter vorstellen. Um die Lampe zum Leuchten zu bringen, müssen der eine UND der andere Schalter eingeschaltet sein. Nachdem die Relais 2, 6 und 7 immer noch aus unserem vorigen Beispiel eingeschaltet sind, wollen wir nun das Relais 2 ausschalten, ohne die anderen zu beeinflussen.

Datenbit	7	6	5	4	3	2	1	0	
	1	1	0	0	0	1	0	0	= 196
UND	1	1	1	1	1	0	1	1	= 251
	1	1	0	0	0	0	0	0	= 192

An alle Stellen, die wir nicht beeinflussen wollen, haben wir eine 1 geschrieben, nur das Bit 2 hat den Wert 0.
In den Positionen, in den beide Bits gesetzt sind, erhalten wir im Ergebnis eine Eins. An allen anderen Stellen ist ja nur ein 'Schalter geschlossen', das Ergebnis ist also null.
In BASIC sieht die Programmzeile dann so aus:

```
POKE 56577,PEEK (56577) AND 251
```

Auch hier haben wir uns mit der PEEK-Anweisung den aktuellen Zustand der Relais gelesen, dann mit der AND-Anweisung das Bit 2 gelöscht und das Ergebnis wieder in die Speicherzelle zurückgeschrieben.
Bleibt nur noch zu erklären, warum an den Stellen 5 bis 3 sowie 0 und 1 ebenfalls die Bits im zweiten Wert (der auch Maske genannt wird) gesetzt sind. Nun, die Erklärung ist, daß nicht immer genau bekannt ist, welche Bits in einem Byte gesetzt sind. Um also jede Möglichkeit auszuschliessen, Bits zu beeinflussen, deren Zustand erhalten bleiben soll, werden in der Maske alle Bits gesetzt, die unbeeinflusst bleiben sollen.

Wem nach so viel Theorie der Kopf raucht, der sollte jetzt zum Kapitel 4.1 weiterblättern. Dort ist eine kleine Schaltung angegeben, mit der 8 Leuchtdioden an den User-Port angeschlossen werden. Damit lassen sich die zuvor beschriebenen Beispiele in der Praxis probieren und vertiefen.

3.0 Anschlußmöglichkeiten am Commodore 64

Der Commodore 64 ist wie kaum ein anderer Computer dieser Preisklasse mit einer Vielzahl der flexibelsten Schnittstellen versehen. Diese Schnittstellen und ein geradezu fantastisches Hardware-Konzept ermöglichen den Anschluß der unterschiedlichsten Selbstbaugeräte.
Dabei sind beim C64 mit geringem Aufwand Dinge möglich, die auch mit ungleich größerem Aufwand an Bauteilen nur schwer oder gar nicht auf anderen Computern realisierbar sind.
Die in jeder Hinsicht leistungsfähigste Anschlußmöglichkeit ist der sogenannte EXPANSION-PORT. Dieser Anschluß ist aber nicht ganz einfach zu handhaben.
Aber auch der USER-PORT ermöglicht mit geringem Hardwareaufwand erstaunliche Dinge.
Die beiden Control-Ports werden den meisten Lesern als Anschlußmöglichkeit für Joysticks und Paddles bekannt sein. Wir werden aber sehen, daß wir diese Ports für unsere Anwendungen umfunktionieren können.
Auch der Cassetten-Port wird von uns einer Untersuchung unterzogen werden. Dabei werden Sie interessante neue Verwendungsmöglichkeiten kennenlernen.
Aber betrachten wir USER-PORT UND EXPANSION-PORT etwas genauer. Beginnen wir mit dem User-Port.

3.1 USER - PORT

Der USER-PORT ist in jeweils etwas unterschiedlichen Belegungen an allen Computern der Firma Commodore zu finden. Das gilt für den legendären PET 2001 genauso wie für den VC-20 oder die neuen Geräte der 600er und 700er Serie.
Der USER-PORT ist die am einfachsten zu handhabende Schnittstelle am C64. Er besteht aus einem 24 poligen Leiterplattenstecker. Einen für diesen Port passenden Stecker dürften Sie bei fast jedem Computerhändler oder Elektronikladen bekommen. Dieser Stecker muß einen Kontaktabstand von 3,96mm aufweisen.
Die Belegung der einzelnen Kontakte ist folgendermaßen :

1 GND	2 +5Volt	3 $\overline{\text{RESET}}$
4 CNT1	5 SP1	6 CNT 2
7 SP2	8 $\overline{\text{PC 2}}$	9 ATN IN
1o 9VAC~	11 9VAC~	12 GND

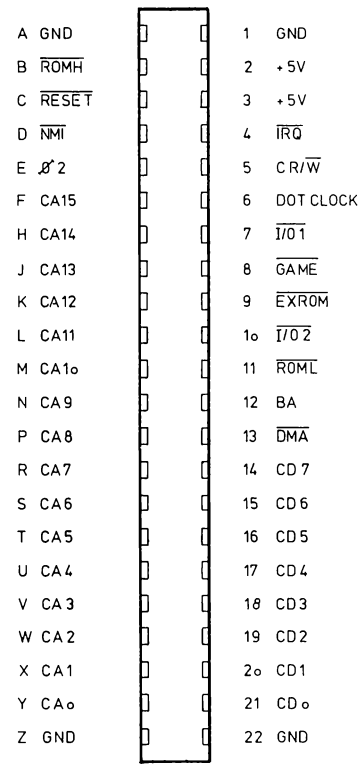
A GND	B $\overline{\text{FLAG 2}}$	C PB o
D PB1	E PB 2	F PB 3
H PB4	J PB 5	K PB 6
L PB7	M PA 2	N GND

Die vier Kontakte 1, 12, A und N sind mit Massepotential des Rechners verbunden. Am Kontakt 2 liegt eine stabilisierte Gleichspannung von +5 Volt. Diese Spannung darf mit maximal 100 Milliampere belastet werden.
Eine Wechselspannung von 9Volt kann an den Kontakten 9 und 10 abgenommen werden.
Der Anschluß 4 ist mit der Reset-Leitung des Prozessors verbunden. Über diesen Anschluß kann der vom Rechner beim Einschalten erzeugte Reset auf externe Schaltungen hinausgegeben werden.
Die Funktion und Verwendung der übrigen Signale wird im Kapitel 4.0 bei der Beschreibung der CIAs 6526 erklärt.

3.2 EXPANSION - PORT

Der EXPANSION-PORT dient in der Regel zur Aufnahme von Spiele-Modulen, BASIC-Erweiterungen oder Ein/Ausgabe-erweiterungen (z.B. IEC-Bus, 80-Zeichen-Karte u.s.w).
An diesem Slot stehen alle wichtigen Prozessor- und Systemsignale zur Verfügung.Durch den speziellen Aufbau des C64 kann sogar ein anderer als der eingebaute 6510-Prozessor auf die Speicherbausteine und die I/O-ICs zugreifen. Von dieser Möglichkeit macht die CP/M-Cartridge Gebrauch.

Die 44 Kontakte dieser Schnittstelle sind wie folgt belegt:



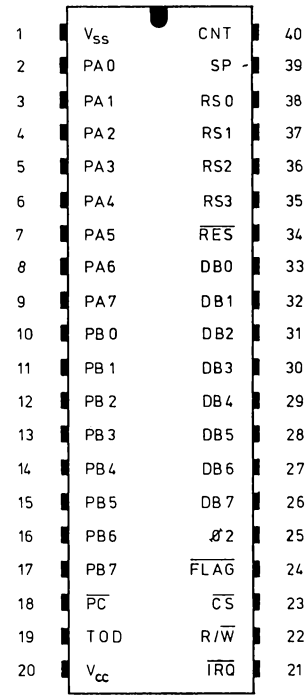
Wie auch beim USER-PORT sind die vier äußeren Kontakte des EXPANSION-PORTs mit Masse verbunden. Die Kontakte 2 und 3 liegen auf +5 Volt Gleichspannung.
Der Adressbus des Prozessors CA0 bis CA15 liegt auf den Kontakten F bis Y. Über diese Leitungen spricht der Prozessor die einzelnen Speicherplätze in RAM, ROM und Peripheriebausteinen an. Bei DMA-Betrieb kann über diese Leitungen ein externer Prozessor oder eine andere Hardware auf den Speicherraum zugreifen.
Der Datenbus CD0 bis CD7 liegt auf den Anschlüssen 14 bis 21. Über diese Leitungen findet der Datentransfer von und zum Speicher statt. Wie auch der Adress-

bus kann dieser Bus von externen Schaltungen genutzt werden. Das Signal -RESET am Anschluß C ist mit dem entsprechenden USER-PORT Anschluß identisch.
02 an Pin E ist das Prozessortaktsignal. Dieses Signal hat eine Frequenz von ca.985 KHz.
Über -IRQ (Pin 4) und -NMI (Pin D) können Interrupts von externen Baugruppen ausgelöst werden.
Auch CR/-W wird auf zwei Arten genutzt. Zum einen liegt an diesem Anschluß das Read/Write-Signal des Prozessors, bei DMA oder Betrieb eines externen Prozessors kann über diese Leitung aber auch der Schreib-Lesevorgang von außen gesteuert werden.
DOT CLOCK am Anschluß 6 ist ein Signal mit einer Frequenz von 7.88 MHz. Diese Frequenz benutzt der VIC-Chip, um die einzelnen Punkte, aus denen ein Zeichen auf dem Bildschirm dargestellt wird, zu erzeugen.
Die beiden -I/O Signale (Pin 7 und Pin 10) sind durch Adressdekodierung gewonnen. Im Adressbereich $DE00 bis $DEFF ist -I/O1 und im Bereich $DF00 bis $DFFF -I/O2 Low.
Die Leitungen -DMA und BA an den Pins 12 und 13 werden in Verbindung mit der DMA-Technik oder dem Betrieb eines zweiten Prozessors benutzt.
-GAME und -EXROM an den Pins 8 und 9 werden benutzt, um die Speicheraufteilung in C64 umzuorganisieren. Wenn -GAME auf Masse liegt, wird das Basic-Rom abgeschaltet. Statt dessen liegt ein externes Rom im Adressbereich von $A000 bis $BFFF. Bei Low an -EXROM wird der Ram-Bereich von $8000 bis $9FFF abgeschaltet. Dafür liegt dann ein externes Rom in diesem Adressbereich.

4.0 Die CIA 6526

Die im C64 eingebauten CIAs (Complex interface Adapter) stellen eine Weiterentwicklung der bekannten VIA 6522 dar. Die wichtigsten Daten des 6526 sind:

- 2 mal 8 Bit-Parallel-I/O-Port
- 2 Handshake-Pins für Parallel-I/O
- 2 kaskadierbare 16-Bit-Timer
- 1 24-Std-Echtzeituhr
- 1 8-Bit-Serial-Port zur seriellen Daten-Ein/Ausgabe



Die CIAs übernehmen im C64 eine Menge von verschiedenen Funktionen. So wird die Tastaturabfrage, die Steuerung des IEC-Busses, Teile der Datasettensteuerung und einiges mehr von den CIAs erledigt.

Die Leitungen mit den Bezeichnungen PA0-7 und PB0-7 stellen die beiden 8-Bit-I/O-Ports dar. Diese Leitungen können wahlweise als Ein- oder Ausgang benutzt werden.
Die CIA-Anschlüsse -FLAG und -PC (Port Control) arbeiten als sogenannte Handshake-Leitungen.
Ein von außen angelegter Low-Pegel an -FLAG signalisiert die Gültigkeit der Daten am Port B.
Dazu wird im Interrupt-Control-Register (ICR) das Bit 4 gesetzt. Damit können die am Port B liegenden Daten in den Speicher gebracht werden.

Entprechend anders herum arbeitet der Anschluß -PC. Hier erscheint ein kurzer Low-Impuls, wenn der Prozessor auf das Portregister B zugreift.
CNT (Count) kann je nach Betriebsart wahlweise als Eingang oder als Ausgang arbeiten. In Verbindung mit den 16-Bit-Zählern arbeitet CNT als Triggereingang, an den die zu zählenden Impulse gelegt werden. In Verbindung mit dem seriellen Schieberegister kann an CNT der Schiebetakt entnommen bzw. eingespeist werden.
An den Anschluß TOD (Time Of Day) wird das Taktsignal für die eingbaute Echtzeituhr gelegt. Diese Echtzeituhr wird von der Netzfrequenz von 50 Hz gesteuert. Dadurch erhält diese Uhr eine ganz ausgezeichnete Langzeitgenauigkeit.
Der Pin SP (Serial Port) ist der Ein/Ausgang des Schieberegisters. Mit Hilfe dieses Anschlusses läßt sich beispielsweise eine RS-232-Schnittstelle für den C64 realisieren.
Die verschiedenen Funktionen der CIA werden über 16 im IC integrierte Register bestimmt. Um die Register ansprechen zu können, sind die Leitungen RS0 bis RS3 vorhanden. Diese Anschlüsse sind mit den 4 untersten Adressleitungen verbunden. Dadurch ist jedes Register wie ein normaler Speicherplatz im C64 anzusprechen.
Die Funktion der einzelnen Register ist teilweise etwas komplex, wir werden sie aber der Reihe nach ausführlich besprechen.
Die Pins DB0-7 sind mit dem Datenbus des Prozessors verbunden. Über diese bidirektionalen Leitungen werden die Daten von bzw. zum Prozessor geführt.
Die übrigen Anschlüsse stellen systeminterne Signale dar und sollen im Moment nicht näher betrachtet werden.

Betrachten wir nun den internen Registeraufbau des 6526.

Reg 0 PRA — Port Register A
Dieser Register stellt beim Lesen den der 8 Leitunngen von Port A dar. Dabei ist es unerheblich, ob die Leitungen als Ein- oder Ausgang programmiert sind.

Reg 1 DDRA — Data Direction Register A
Der Inhalt dieses Registers bestimmt, ob die Leitungen des Ports A als Ausgang oder als Eingang arbeiten. Ist in diesem Register ein Bit gesetzt, dann arbeitet das entsprechende Portbit als Ausgang, ein gelöschtes Bit programmiert das Portbit als Eingang.

Reg 2 PRB — Port Register B
Arbeitet genau wie PRA für den Port B.

Reg 3 DDRB — Data Direction Register B
Arbeitet genau wie DDRA für den Port B.

Reg 4 TA LO — Timer A Low Byte
Ein Lesen dieses Registers liefert den momentanen Zustand des niederwerigen Bytes des 16 Bit Timers A.
Beim Schreiben in dieses Register wird es mit dem niederwertigen Byte des 16 Bit Wertes geladen, von dem der Zähler nach Null zählt.

Reg 5 TA HI — Timer A High Byte
Ein Lesen dieses Registers liefert den momentanen Zustand des höherwerigen Bytes

des 16 Bit Timers A.
Beim Schreiben in dieses Register wird es mit dem höherwertigen Byte des 16 Bit Wertes geladen, von dem der Zähler nach Null zählt.

Reg 6 TB LO — Timer B Low Byte
Arbeitet genau wie TA LO für Timer B.

Reg 7 TB HI — Timer B High Byte
Arbeitet genau wie TA HI für Timer B.

Reg 8 TOD 10THS — Time Of Day (1/10 Sekunde)
Von diesem Register werden nur die Bits 0 bis 3 benutzt.
Beim Lesen dieses Registers erhält man die 1/10 Sekunden der Echtzeit-Uhr im BCD-Format. Es können also nur die Zahlen von 0 bis 9 auftreten.
Wird in dieses Register geschrieben, so muß unterschieden werden, wie Bit 7 im Register 15 gesetzt ist. Ist dies Bit 0, dann wird durch Schreiben die Uhrzeit gesetzt, ist das Bit 1, dann wird die Alarmzeit gesetzt.
Die Bits 4 bis 7 dieses Registers sind immer Null.

Reg 9 TOD SEC — Time Of Day (Sekunden)
Die Bits 0 bis 3 dieses Registers liefern beim Lesen die Sekunden der Uhrzeit im BCD-Format, die Bits 4 bis 6 die Zehnersekunden. Bit 7 ist immer Null.
Das Beschreiben dieses Registers ist analog zum Reg 8.

Reg 10 TOD MIN — Time Of Day (Minuten)
Die Bits 0 bis 3 dieses Registers liefern beim Lesen die Minuten der Uhrzeit im BCD-Format, die Bits 4 bis 6 die Zehnerminuten. Bit 7 ist immer Null.
Das Beschreiben dieses Registers ist analog zum Reg 8.

Reg 11 TOD HR — Time OF Day (Stunden)
Die Bits 0 bis 3 dieses Registers liefern beim Lesen die Stunden der Uhrzeit im BCD-Format, das Bit 4 die Zehnerstunde. Die Bits 5 und 6 sind immer 0, das Bit 7 signalisiert bei 0 AM, also vormittags, und bei 1 PM, nachmittags.
Das Beschreiben dieses Registers ist analog zum Reg 8.

Reg 12 SDR — Serial Data Register
Dieses Register ist ein Schieberegister, über das Daten sowohl über den Pin SP herausgeschoben als auch über SP hineingeschoben werden können. Wird SP als Ausgang verwendet, so erscheint das höchstwertige Bit zuerst.

Reg 13 ICR — Interrupt Control Register
Dieses Register signalisiert verschiedene Zustände der CIA und kann beim Auftreten bestimmter Zustände einen Interrupt an den Prozessor schicken.
Beim Lesezugriff werden die Daten aus dem INT DATA Register gelesen.
Beim Lesen dieses Registers werden alle Bits

gelöscht!
Die einzelnen Bits dieses Registers haben folgende Funktion:

Bit 0 1 = Unterlauf von Timer A

Bit 1 1 = Unterlauf von Timer B

Bit 2 1 = Uhrzeit und Alarmzeit sind gleich

Bit 3 1 = SDR ist voll oder leer, je nach Betriebsart.

Bit 4 1 = negative Flanke an Pin FLAG aufgetreten

Bit 5 immer 0

Bit 6 immer 0

Bit 7 1 = Übereinstimmung von mindestens einem gesetzten Bit in INT MASK und INT DATA
Beim Schreiben in das ICR kommen die Daten in das INT MASK Register. Aus dem INT MASK Register kann nicht gelesen werden.
Die Bedeutung der einzelnen Bits ist identisch mit den Bedeutungen im INT DATA Reg. Ausnahme ist das Bit 7.
Ist es gesetzt, so setzt jedes 1-Bit das korrespondierende MASK-Bit. Die anderen Bits werden nicht beeinflusst.
Ist es gelöscht, dann löscht jedes 1-Bit das korrespondierende MASK-Bit. Auch hierbei werden die überigen Bits nicht beeinflusst.
Ist im INT MASK ein Bit gesetzt und wird das korrespondierende INT DATA Bit gesetzt, dann wird ein Low-Signal am Pin -IRQ erzeugt.

Reg 14 CRA Control Register A
Dieses Register bestimmt verschiedene Betriebsarten der CIA. Auch bei diesem

Register sind die einzelnen Bits funktionsbestimmend.

Bit 0 1 = Timer A startet
0 = Timer A stoppt

Bit 1 1 = jeder Unterlauf von Timer A wird an PB6 signalisiert. Diese Funktion hat Vorrang vor der Festlegung im DDRB.

Bit 2 1 = tritt ein Unterlauf im Timer A auf, so wird PB6 in die entgegengesetzte Polarität geschaltet.
0 = tritt ein Unterlauf im Timer A auf, so wird PB6 für die Dauer eines Systemtaktes (ca. 1 microsek) High.

Bit 3 1 = Timer A zählt nur einmal vom geladenen Wert auf Null und hält dann an.(One-Shot-Mode).
0 = Timer A zählt fortlaufend vom geladenen Startwert nach Null und beginnt selbsttätig von vorn.(Continuous-Mode)

Bit 4 1 = unbedingtes Laden eines neuen Zählerstartwertes. Dieses Bit erlaubt das Laden des Zählers zu jedem beliebigen Zeitpunkt.

Bit 5 1 = der Timer A zählt positive Flanken am Pin CNT. Dadurch kann der Timer/Counter auch externe Impulse zählen.
0 = Der Timer A Systemtaktpulse.

Bit 6 1 = SP arbeitet als serieller Eingang
0 = SP ist Ausgang des SDR

Bit 7 1 = TOD Echtzeituhr arbeitet mit einer Netzfrequenz von 50 Hz.
0 = TOD arbeitet mit 60 Hz (üblich in den USA und anderen Ländern)

Reg 15 CRB Control Register B

Auch CRB ist wieder in einzelnen Bits zu betrachten.

Bit 0-4 wie Reg 14 Bit 0-4 für Timer B und PB7

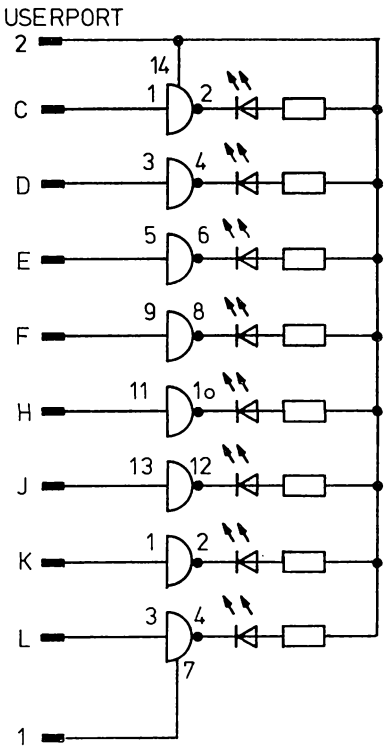
Bit 5-6 diese Bits bestimmen die Quelle des Trigger für Timer B. Dabei gilt:
00 = Timer B zählt Systemtakte
01 = Timer B zählt positive Flanken an CNT
10 = Timer B zählt Unterläufe von Timer A
11 = Timer B Zählt Unterläufe von A, wenn CNT = High ist

Bit 7 1 = TOD Alarm Register beschreiben
0 = TOD Uhrzeit Register beschreiben

Im C64 sind zwei dieser famosen Bausteine eingebaut. Wie bereits besprochen, sind sie wie alle Periperie-Bausteine im C64 Memory-Mapped, d.h. genau wie Speicherplätze anzusprechen. Die Basisadressen der CIAs sind $DC00 (56320) für CIA1 und $DD00 (56576) für CIA2.
Am User-Port haben wir den ganzen Port B der CIA 2 sowie verschiedene andere Anschlüsse von CIA1 und CIA2 zu freier Verfügung. Mit diesen Leitungen wollen wir nun unsere ersten Experimente beginnen.

4.1 LEDs am USER - PORT

Die Stromversorgung für die unten abgebildete Schaltung können Sie unbedenklich dem C64 an den Kontakten 1 und 2 abnehmen. Die gesamte Stromaufnahme liegt auch bei eingeschalteten LEDs unter 100 mA.



Alle Widerstände
330Ω

Gatter 2×7406

bei beiden IC's
Pin 14 mit + und
Pin 7 mit -
verbinden.

Das verwendete IC 7416 enthält 6 identische invertierende Pufferstufen mit Open-Collector-Ausgängen. Darum werden 2 ICs für 8 LEDs benötigt. Laut Datenblatt vermag ein Ausgang einen Strom von min. 40 mA zu schalten. Durch die Vorwiderstände vor den LEDs wird der Strom aber auf einen Wert von weniger als 10 mA begrenzt.

Um eine bestimmte LED einzuschalten, benutzen wir folgendes kleines Programm:

```
10 CI = 56576
20 PB = CI + 1
30 DR = CI + 3
40 POKE DR , 255
50 INPUT "WELCHE LED (1-8)";LE
60 IF LE < 1 OR LE > 8 THEN END
70 POKE PB , 2 ↑ (LE - 1)
80 GOTO 50
```

Betrachten wir dieses kleine Programm einmal Zeile für Zeile.

10 - 30 Der Variablen CI wird die Startadresse der CIA2 ($DD00) zugewiesen. PB enthält die Adresse des Ports B, DR die des Datenrichtungsregisters.

40 In dieser Zeile wird das Datenrichtungsregister so programmiert, das alle Leitungen von Port B als Ausgang arbeiten.

50 Mit der INPUT-Anweisung wird die Position der gewünschten LED eingegeben. Der Wert darf zwischen 1 und 8 liegen.

60 Anschließend wird geprüft, ob der eingegebene Wert auch zulässig ist. Ist der Wert zu groß oder klein, wird das Programm beendet.

70 In dieser Zeile wird nun der dezimale Wert errechnet, der in den Port B gePOKEd wird. Damit wird die entsprechende LED eingeschaltet.

80 Abschliessend wird zur Zeile 50 zurückgesprungen. Sie können nun eine neue LED angeben oder das Programm beenden.

Dieses kleine Programm hat einen Schönheitsfehler. Es kann zur Zeit nur eine LED eingeschaltet sein. Sobald eine neue LED angegeben wird, verlischt die zuvor Eingeschaltete. Diesen Fehler behebt das nächste Programm.

```
 10 CI = 56576
 20 PB = CI + 1
 30 DR = CI + 3
 40 POKE DR , 255
 50 INPUT "WELCHE LED (1-8)";LE
 60 IF LE < 1 OR LE > 8 THEN END
 70 INPUT "AN (1) ODER AUS (0)";AN
 80 AW = PEEK (PB)
 90 IF AN = 0 THEN POKE PB , AW AND (255 - (2 ↑ (LE - 1)))
100 IF AN = 1 THEN POKE PB , AW OR (2 ↑ (LE - 1))
110 GOTO 50
```

Bis einschließlich Zeile 60 ist dieses Programm identisch mit unserem ersten Programm. Beginnen wir also gleich bei der nächsten Zeile.

70 In dieser Zeile erfolgt die Abfrage, ob die eingegebene LED ein- oder ausgeschaltet werden soll. Dazu wird der Variablen AN der Wert 1 übergeben, wenn eine LED eingeschaltet werden soll, sonst ist AN = 0.

80 In die Variable AW wird der Zustand des Ports B

gespeichert. Dieser Wert gibt an, welche LEDs eingeschaltet sind.

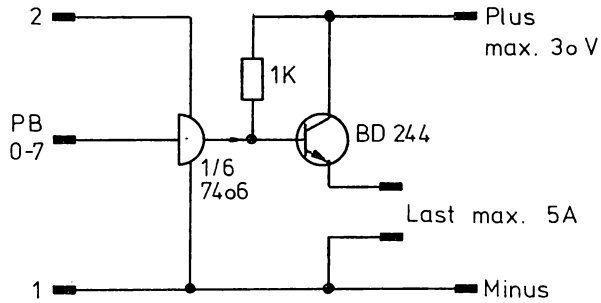
90 Ist AN = 0, dann wird der Wert AW mit dem Klammerausdruck UND-verknüpft. Dieser Klammerausdruck ergibt eine Zahl, in der alle bis auf das zu löschende Bit gesetzt sind.

100 Ist AN = 1, dann wird der Wert AW mit dem Klammerausdruck verODERt. Dadurch wird zusätzlich zu den schon gesetzten Bits das gewünschte Bit gesetzt, die gewählte LED leuchtet.

110 Abschließend wird zur Zeile 50 gesprungen, und Sie können eine neue LED ein- oder ausschalten.

4.2.0 Schalten mit Transistor

Die CIA kann an den Ausgängen nur Ströme von wenigen mA liefern. Damit kann nicht einmal eine Leuchtdiode betrieben werden. Darum haben wir in Kapitel 4.1 Puffer-ICs verwendet, um einen größeren Strom zu schalten. Da aber ein Strom von 40 mA für viele Anwendungen immer noch zu gering ist, nachfolgend eine Schaltstufe für maximal 5 Ampere. Bei so großen Schaltströmen kann allerdings ein kleiner Kühlkörper nötig sein, um die entstehende Verlustwärme des Transistors abzuführen.



Natürlich ist die Stromversorgung des C64 für so hohe zusätzliche Leistungen nicht konzipiert. Darum muß ein externes Netzteil die benötigten Ströme liefern. Dabei sollte die maximale Spannung am Kollektor des Transistors 30 Volt nicht übersteigen.

An dieser Stelle ist eine Warnung angebracht. Sollten Sie versehentlich mit einer Spannung über 5 Volt direkt an die User-Port-Anschlüße geraten, so bedeutet dies das entgültige Aus für Ihre CIA. Da dieser Baustein aber schwer erhältlich

und zudem recht teuer ist, sollten Sie schon ganz sicher sein, bevor Sie mit höheren Spannungen arbeiten oder aber dieses Risiko elegant mit einem Relais oder einem Optokoppler umgehen.

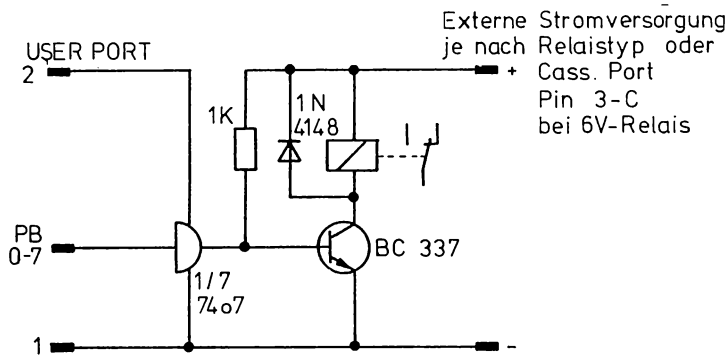
Diese Möglichkeiten wollen wir in den beiden nächsten Kapiteln besprechen.

4.2.1 Schalten mit Relais

Obwohl in der Computertechnik nicht sehr häufig eingesetzt sind Relais auch heute noch interessante Bauteile mit einigen Vorteilen vor elektronischen Lösungen. So ist der Kontaktwiderstand in der Regel sehr gering. Außerdem sind über die Kontakte Wechselspannungen problemlos schaltbar.
Die Baugröße moderner Relais ist überraschend gering. So gibt es mittlerweile Bauformen, die einem 14-poligen IC entsprechen.
Leider sind Relais für 5 Volt nur schwer erhältlich. Die 6 Volt-Typen tun sich aber beim Betrieb mit nur 5 Volt vereinzelt etwas Schwer. Glücklicherweise verfügt der C64 über einen recht leistungsstarken 6 Volt-Ausgang. Dieser Ausgang ist sogar stabilisiert und abschaltbar!
Gemeint sind die Kontakte 3 und C des Cassetten-Ports. Über diese Kontakte wird normalerweise der Motor eines Datenrecorders gesteuert.
Gesteuert wird dieser Ausgang vom Bit 5 des im Prozessor 6510 integrierten Parallel-Ports. Ist dieses Bit gesetzt, dann ist der Motorausgang abgeschaltet. Um also ein Gerät zu betreiben, muß das Bit 5 im Prozessorport gelöscht werden.
Dieser Prozessorport belegt die Adressen 0 und 1 in der Zero-Page. Ähnlich wie die Ports in den CIAs besteht auch dieser Port aus einem Datenrichtungsregister und dem

eigentlichen Datenregister. Das DDR belegt die Adresse 0, während der Datenport die Adresse 1 belegt. Da die Bits 0 bis 2 des Prozessorports für die Speicherverwaltung benutzt werden, ist beim Ändern der Inhalte der beiden Speicherplätze eine gewisse Vorsicht angebracht. An sonsten wird sich Ihr C64 verabschieden, d.h. auch mit RUN STOP/RESTORE nicht wiederbeleben lassen. Da hilft dann nur noch aus- und wieder einschalten.

Handelsübliche 6 Volt-Relais benötigen einen Nennstrom von ca. 60 bis 100 mA. Da der Cassettenmotorausgang maximal 500 mA liefert, können je nach Nennstrom bis zu 8 Relais ohne zusätzliche Stromversorgung betrieben werden.



Wenn der Eingang der Schaltstufe mit einem Ausgang des Parallel-Ports verbunden ist, dann zieht das Relais bei gesetztem Portbit an.
Um aber überhaupt die Relais schalten zu können, muß zuerst der Cassettenmotorausgang eingeschaltet werden.
Das nachstehende Programm schaltet ein von PB0 gesteuertes Relais abwechselnd ein und aus.

```
 10 CI = 56576
 20 PB = CI + 1
 30 DR = CI + 3
 40 DP = 0
 50 PP = 1
 60 POKE DP , 63
 70 POKE PP , 7
 80 POKE DR , 255
 90 POKE PB , 0
100 PRINT "DAS RELAIS IST AUS"
110 PRINT "DRUECKEN SIE EINE TASTE FUER ENDE"
120 FOR I = 1 TO 1000
130 GET A$ : IF A$ <> "" THEN GOTO 220
140 NEXT I
150 POKE PB , 1
160 PRINT "DAS RELAIS IST AN"
170 PRINT "DRUECKEN SIE EINE TASTE FUER ENDE"
180 FOR I = 1 TO 1000
190 GET A$ : IF A$ <> "" THEN GOTO 220
200 NEXT I
210 GOTO 90
220 POKE DP , 47
230 POKE PP , 55
240 POKE PB , 0
250 PRINT "JETZT IST ALLES AUS"
```

Die Variablen CI, PB und DR sind bereits aus den vorherigen Beispielen bekannt. Beginnen wir also bei der Programmzeile

40 - 50 Die Variable DP enthält die Adresse des Datenrichtungsregisters des Prozessors, PP die Adresse des Prozessorports selber.

60 Alle Leitungen des Prozessorports werden auf Ausgang programmiert.

70 Die Motorspannung wird eingeschaltet.

80 Die Datenrichtung von Port B der CIA wird auf Ausgang festgelegt.

90 Alle Bits von Port B auf 0. Damit ist jetzt die Versorgungsspannung für das Relais eingeschaltet, das Relais ist aber noch nicht angezogen.

100 - 110 Auf dem Bildschirm wird der Zustand des Relais angezeigt.

120 - 140 Die in diesen drei Zeilen programmierte Zeitschleife fragt zusätzlich die Tastatur auf Tastendrücke ab. Ist eine Taste gedrückt, so wird zur Zeile 220 verzweigt.

150 Das Bit 0 des Ports B wird gesetzt, damit ist das Relais gesetzt.

160 - 200 Wie die Zeilen 100 - 140

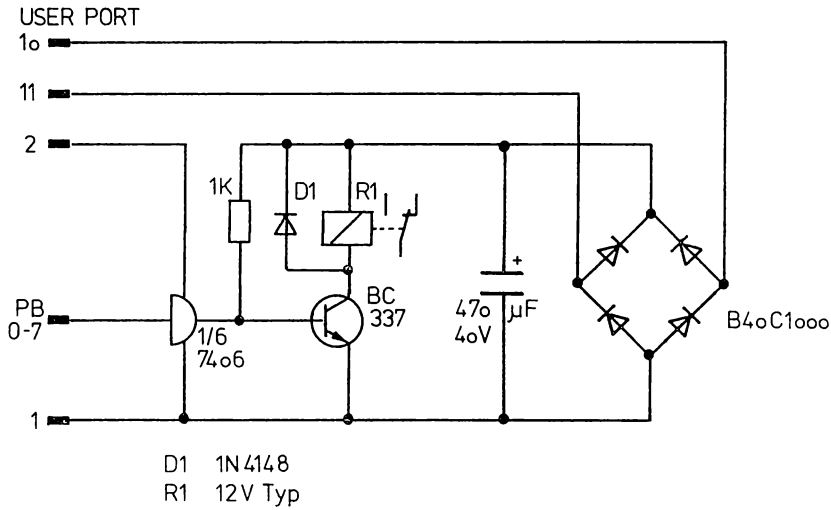
210 Das Programm springt in die Zeile 90 zurück, das Relais wird wieder ausgeschaltet und ein neuer Durchlauf beginnt.

220 - 250 Wurde ein Tastendruck festgestellt, springt das Programm in diese Zeilen. Hier werden Prozessordatenrichtungsregister und Prozessorport wieder mit den ursprünglichen Werten geladen, Port B wird abgeschaltet und das Programm wird beendet.

Wenn Sie nur ein Relais benötigen, kann natürlich das Relais auch nur über den Cassettenport gesteuert werden. In diesem Fall entfällt dann die zusätzliche Transistorstufe.

Unabhängig von der Art der Steuerung des Relais sollten Sie die eingezeichnete Diode einsetzen. Beim Abschalten des Relais entstehen hohe Induktionsspannungen (je nach Typ mehrere 100 Volt!), die über die sogenannte Freilaufdiode kurzgeschlossen werden.

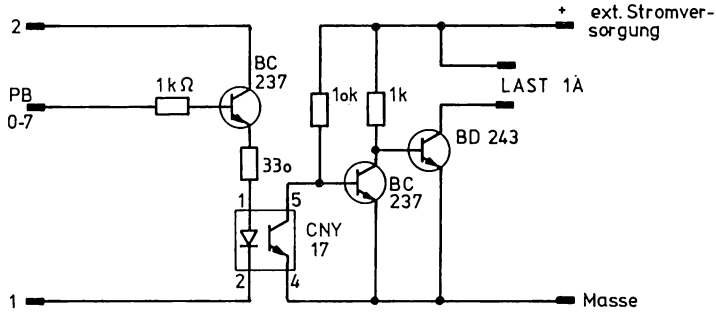
Aber auch Relais für eine Beriebsspannung von 12 Volt lassen sich ohne externe Stromversorgung am C64 betreiben.
An den Kontakten 10 und 11 des User-Ports steht eine Wechselspannung von 9 Volt zur Verfügung. Nach Gleichrichtung und Siebung dieser Spannung erhalten wir eine Gleichspannung von ca. 12 Volt. Der maximale hier entnommene Strom ist im C64-Handbuch mit 100mA angegeben. Diese Angabe hat aber nur Gültigkeit bei gleichzeitigem Betrieb der Datasette. Ohne Datasette kann hier ohne Bedenken ein Strom von 300 bis 400 mA entnommen werden. Damit lassen sich maximal 8 Relais versorgen.
Wenn Sie die 9 Volt Wechselspannung einmal überlasten, z.B. durch einen versehentlichen Kurzschluß, dann brennt eine im C64 eingebaute Schmelzsicherung durch. Diese Sicherung hat einen Wert von 1,25 Ampere. Nur in einigen älteren C64 befindet sich eine Sicherung mit einem Wert von 1 Ampere. Diese kann bei Bedarf gegen eine 1,25 A-Sicherung getauscht werden.



D1 1N4148
R1 12V Typ

4.2.2 Ansteuerung von Optokopplern

Die im vorigen Kapitel besprochene Potentialtrennung mit Relais ist für viele Anwendungen nicht einsetzbar. Einmal ist der erforderliche Strom für die Relais nicht immer ohne weiteres verfügbar. Außerdem ist die erreichbare Schaltfrequenz sehr niedrig.
Der Einsatz von Optokopplern ist immer dann empfehlenswert, wenn mit einer höheren Spannung als 5 Volt gearbeitet werden soll. Durch die vollständige galvanische Trennung zwischen Computer und Peripherie ist der Computer optimal geschützt.
Die abgebildete Schaltung ist eine Modifikation der zuvor beschriebenen Transistorschaltstufe.
Die Ansteuerseite des Optokopplers besteht aus einer einfachen Transistorstufe. Wenn das Portbit gesetzt ist, dann fließt ein Strom duch die im Koppler integrierte LED. Dadurch wird der Phototransistor durchgesteuert und der Transistor T1 sperrt. Damit wird die Basis des T2 angesteuert, T2 schaltet durch.
Die maximale Spannung im Lastkreis richtet sich nach der Spannungsfestigkeit des Transistors im Optokoppler und des verwendeten Leistungstransistors.
Der Phototransistor des im Beispiel eingesetzten CNY 17 verträgt eine maximale Kollektorspannung von 70 Volt, andere ebenfalls einsetzbare Typen wie z.B der IL-74 haben zulässige Kollektorspannungen von etwa 20 Volt.
Der eingesetzte Leistungstransistor BD 243 verträgt eine maximale Kollektorspannung von 60 V und einen Schaltstrom von 6 Ampere, der Transistor BC 237 immerhin 45 V. Damit ist die maximale Betriebsspannung der Stufe in der angegebenen Dimensionierung durch den Transistor T1 bestimmt.
Beim Entwurf einer Schaltung mit anderen Optokopplern und Transistoren entnehmen Sie diese Maximalwerte bitte den Datenblättern der verwendeten Bauteile.

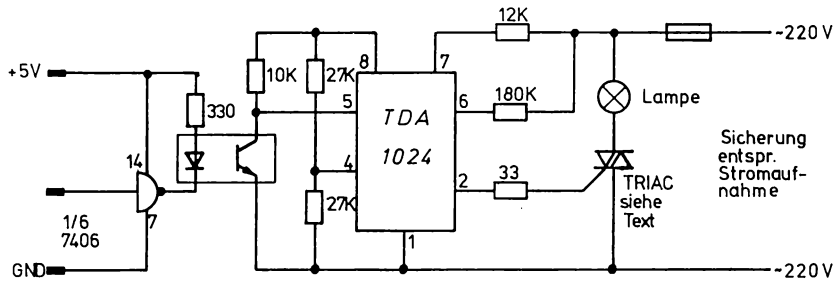


4.2.3 TRIAC-Schaltstufe mit Optokopplern

Zum Abschluß dieses Kapitels eine kleine Schaltung zum Steuern eines 220 Volt-Verbrauchers über einen Optokoppler und einen TRIAC.
Diese Schaltung, wie übrigens alle mit 220 Volt betriebenen Schaltungen sind nur für versierte Bastler geeignet. Beachten Sie unbedingt die nötigen Sicherheitsmassnahmen. Arbeiten Sie an diesen Schaltungen nur bei ausgeschaltetem Gerät, besser noch, ziehen Sie den Netzstecker heraus. Isolieren Sie alle 220 Volt führenden Leitungen und verwenden Sie geschlossene Kunststoffgehäuse. Sehen Sie unbedingt Zugentlastungen für Netzleitungen vor.

220 Volt können TÖDLICH sein !

Die gesamte Schaltung besteht aus einem Optokoppler, dem TRIAC, einigen Widerständen und einem TDA 1024. Das IC enthält alle benötigten Stufen, um einen sogenannten Null-Spannungsschalter aufzubauen. Dabei wird der TRIAC angesteuert, wenn die Netzwechselspannung in der Nähe der Nulldurchgänge liegt. Auf diese Weise werden die beim Bertrieb eines TRIACs sonst entstehenden Störspannungen wirkungsvoll reduziert.
Über einen Vorwiderstand läßt sich der TDA 1024 direkt an 220 Volt betreiben. Dieser Vorwiderstand muß aber mindestens mit 4 Watt belastbar sein.
Als TRIAC kann jeder handelsübliche Typ mit ausreichender Spannungsfestigkeit eingesetzt werden. Die von uns verwendeten Typen TIC 206 (400 V, 3 A) und TIC 226 (400 V, 8 A) sind gut erhältlich und auch recht preiswert.



Beim Betrieb dieser Stufe über den User-Port ist die Programmierung identisch mit der der vorangegangenen Stufen. Sobald die LED im Optokoppler angesteuert wird, schaltet der TRIAC durch.

4.2.4 Lichtorgel mit dem C64

Wer an Lichtorgeln und anderen Lichteffekten seinen Spaß hat, kann mit mehreren TRIAC-Stufen wirklich tollen Lampenzauber erzeugen.
Da die TRIAC-Stufen nicht ganz preiswert sind, können Sie die beschriebenen Programme erst einmal mit LEDs am User-Port probieren.
Bei allen Programmen gehen wir von 8 Lampenstufen aus. Bei Bedarf können aber auch weniger, mit zusätzlichen Peripherie-ICs wie 6522 oder auch 8255 aber auch mehr Lampen angeschlossen werden. Natürlich müssen dann die Programme den geänderten Gegebenheiten angepasst werden.
Das erste Programm erzeugt ein Lauflicht. Die Geschwindigkeit kann mit den Funktionstasten F1 und F3 eingestellt werden.

```
100 C2 = 56576
110 PB = C2 + 1
120 DR = C2 + 3
130 LA = 0
140 T = 100
150 FOR I = 0 TO 7
160 LA = 2↑I
170 POKE PB , LA
180 GET A$
190 IF A$ =CHR$(133) THEN T=T-10: IF T<0 THEN T=0
200 IF A$ =CHR$(134) THEN T=T+10
210 FOR TL = 0 TO T: NEXT TL
220 NEXT I
230 GOTO 150
```

Durch Ändern der Programmzeile 160 können wir zwei aufeinander zulaufende Lichtpunkte erzeugen. Dieser Effekt

ist besonders wirkungsvoll, wenn 16 Lampen angesteuert werden. Aber auch mit 8 Lampen sieht das ganze schon ganz gut aus

```
160 LA = 2↑I + 2↑7-I
```

Auch sehr schön macht sich das folgende kurze Programm. Hierbei werden zuerst alle Lampen der Reihe nach eingeschaltet. Danach werden sie beginnend bei der zuerst eingeschalteten wieder ausgeschaltet. Darauf wiederholt sich der Vorgang, diesmal werden die Lampen aber von der anderen Seite her ein- und wieder ausgeschaltet.

```
100 C2 = 56576
110 PB = C2 + 1
120 DR = C2 + 3
130 LA = 0
140 T = 100
150 FOR I = 0 TO 7
160 LA = LA + 2↑I
170 POKE PB , LA
180 FOR TL = 0 TO T: NEXT TL
190 FOR I = 0 TO 7
200 LA = LA - 2↑I
210 POKE PB , LA
220 FOR TL = 0 TO T: NEXT TL
230 FOR I = 7 TO 0 STEP -1
240 LA = LA + 2↑I
250 POKE PB , LA
260 FOR TL = 0 TO T: NEXT TL
270 FOR I = 7 TO 0 STEP -1
280 LA = LA - 2↑I
290 POKE PB , LA
300 FOR TL = 0 TO T: NEXT TL
```

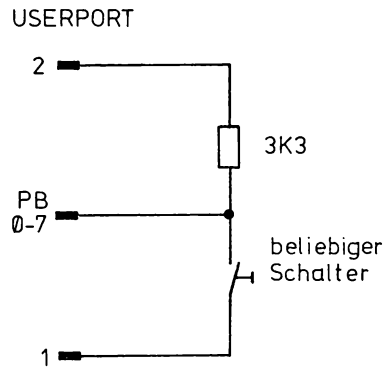
```
310 GET A$
320 IF A$ =CHR$(133) THEN T=T-10: IF T<0 THEN T=0
330 IF A$ =CHR$(134) THEN T=T+10
340 FOR TL = 0 TO T: NEXT TL
350 NEXT I
360 GOTO 150
```

Als letztes im Kapitel über die Datenausgabe über den User-Port wollen wir eine besondere Art der 'Datenausgabe' betrachten. Die Daten, die wir an unsere 'Lichtorgel' senden, sind das Betriebssystem des C64.

```
100 C2 = 56576
110 PB = C2 + 1
120 DR = C2 + 3
130 RA = 57344
140 RE = 65535
140 T = 100
150 FOR I = RA TO RE
160 LA = PEEK (I)
170 POKE PB , LA
180 GET A$
190 IF A$ =CHR$(133) THEN T=T-10: IF T<0 THEN T=0
200 IF A$ =CHR$(134) THEN T=T+10
210 FOR TL = 0 TO T: NEXT TL
220 NEXT I
230 GOTO 150
```

4.3 Eingabe von Daten in den USER - PORT

Bis zu diesem Punkt haben wir nur Daten aus dem Computer über den User-Port ausgegeben. Durch eine entsprechende Programmierung des Datenrichtungsregisters läßt sich aber auch der umgekehrte Weg beschreiten.



Im einfachsten Fall besteht eine Dateneingabe aus der Abfrage eines Tasters.
Wenn der Taster nicht betätigt ist, liegt der User-Porteingang auf High-Pegel. Bei gedrückter Taste erhalten wir ein Low-Signal. Zur Anzeige von 8 am User-Port angeschlossenen Tastern können Sie dieses Programm verwenden.

```
 10  CI = 56576
 20  PB = CI + 1
 30  DR = CI + 3
 40  POKE DR , 0
 50  W = PEEK ( PB )
 60  FOR I = 7 TO 0 STEP -1
 70  PRINT CHR$( 49 + ( X = W AND 2 ↑ I ));
 80  NEXT I
 90  PRINT
100  GOTO 50
```

Betrachten wir das Programm kurz Zeile für Zeile.

10 - 30 Den Variablen CI, PB und DR werden die Adressen

für CIA2, Port B und DDRB zugewiesen.

40 Das Datenrichtungsregister wird programmiert für Port B alle Bits auf Eingang.

50 Der Zustand des Ports B wird mit PEEK gelesen und der Variable W zugewiesen

60 - 80 In diesen drei Zeilen wird der in W stehende Wert in eine Binärzahl gewandelt und ausgedruckt. Ist das Bit mit der Wertigkeit 2↑I auch im Wert W gesetzt, so ist X = 0 und es wird eine 1 gePRINTet. Ist das Bit nicht gesetzt, dann ergibt X = -1, es wird eine 0 gePRINTet.

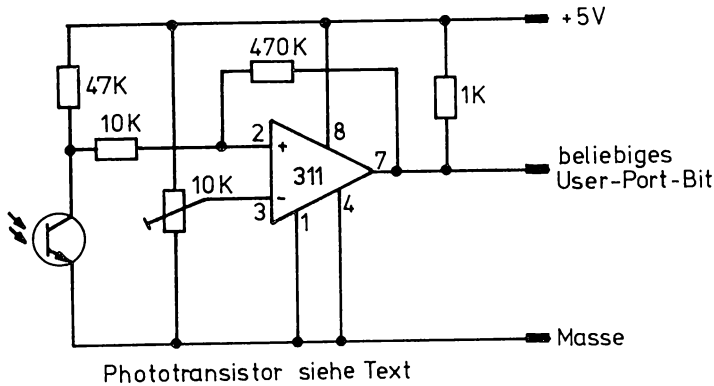
90 - 100 Die Darstellung eines neuen Wertes wird vorbereitet und es wird zur Zeile 50 gesprungen.

Die Zeilen 60 bis 80 stellen eine universell verwendbare Routine zur Zerlegung einer Zahl in ihr Bitmuster dar. Um eine 16-Bit-Zahl zu zerlegen, muß nur die FOR-NEXT-Schleife von 15 bis 0 erweitert werden.

An die Stelle der Schalter kann nun auch der Ausgang eines anderen ICs, ein Transistor oder ein Optokoppler geschaltet werden. Damit lassen sich direkt elektrische Low-High-Pegel von der CIA abfragen. Diese Möglichkeiten werden wir in den nächsten Kapiteln beschreiben.
Nun ist das Abfragen von Schaltern und Darstellen auf dem Bildschirm noch keine aufregende Sache. Trotzdem gibt es aber auch für so eine einfache Angelegenheit wie die zuvor Beschriebene eine mögliche Anwendung.
Die acht Schalter könnten in Ihrer Wohnung die Fenster und die Eingangstür überprüfen. Bei geschickter Verteilung der Schalter und einem etwas umfangreicheren Programm könnte das Bitmuster sofort signalisieren, in welchem Raum welches Fenster oder welche Tür nicht ordnungsgemäß verschlossen ist.

4.4 Die Lichtschranke

Dargestellt ist eine einfache lichtempfindliche Schaltung. Im Wesentlichen besteht die Schaltung aus dem Komparator 311 und einem Phototransistor. Wenn nun auf den Phototransistor Licht fällt, so beginnt er zu leiten, genau so, als würde die Basis des Transistors angesteuert. Dadurch sinkt die Spannung am nicht invertierenden Eingang des Komparators. Mit dem Poti wird am invertierenden Eingang eine Referenzspannung eingestellt.



Liegt die Spannung über dem Phototransistor über der Referenzspannung, also bei geringer Helligkeit, dann ist der Ausgang der Stufe High, wird der Transistor aber ausreichend beleuchtet, dann ist am Ausgang ein Low zu messen.
Als einziger Punkt bleibt noch der Widerstand zwischen dem "+"-Eingang und dem Ausgang. Über diesen Widerstand bekommt die Schaltung eine sogenannte Hysterese. Diese Hysterese ist notwendig, wenn Sie die Schaltung in Räumen mit Leuchtstoffröhren betreiben. Obwohl mit dem Auge nicht

sichtbar, pulsiert das Licht, die Helligkeit nimmt im Takt mit der Netzfrequenz zu und ab. Diese Helligkeitsunterschiede werden vom Phototransistor aufgenommen und würden bei zu geringer Hysterese zu einem ständigen Hin- und Herschalten des Ausgangs führen. Sollten Sie trotzdem Schwierigkeiten mit dem beschriebenen Phänomen haben, können Sie den Widerstand verkleinern. Dadurch ist dann aber zwangsläufig eine größere Helligkeitsschwankung nötig um den Transistor umzuschalten. Ist diese größere Schwankung der Helligkeit nicht realisierbar (z.B. weil die Umgebungshelligkeit zu groß ist), dann hilft in der Regel ein Kondensator von ca. 470 nF parallel zum Phototransistor.

Der Ausgang der Schaltung kann nun auf ein beliebiges Bit des Ports B am User-Port gelegt werden.

Bei geschickter Montage ist damit die Überprüfung von Räumen möglich. Eine andere wichtige Möglichkeit ist das Zählen von Gegenständen.

Die dafür notwendige Software wollen wir im nächsten Kapitel besprechen.

4.4.1 Zählen von Impulsen mit Lichtschranke

Das Zählen von Impulsen, Thema dieses Kapitels, läßt sich natürlich nicht nur mit der Lichtschranke durchführen. Auch ein einfacher Schalter oder eine andere als die vorgeschlagene Elektronik kann angeschlossen werden. Dabei ist nur zu berücksichtigen, daß die Spannung am User-Port 5 Volt nicht übersteigen darf.

Aber wenden wir uns nun der Software zu. Dazu beziehen wir uns auf die Lichtschranke des vorigen Kapitels, die am Bit 0 des User-Ports angeschlossen ist.

Zuerst wollen wir nur einmal fortlaufend alle Impulse zählen.

```
 10 CI = 56576
 20 PB = CI + 1
 30 DR = CI + 3
 40 Z = 0
 50 POKE DR , 254
 60 POKE PB , 0
 70 WE = PEEK (PB)
 80 IF WE = 0 THEN GOTO 70
 90 Z = Z + 1
100 PRINT CHR$(147) ; Z
110 WE = PEEK (PB)
120 IF WE = 1 THEN GOTO 110
130 GOTO 60
```

Wie bisher definieren wir zu Anfang des Programms die benötigten Variablen.

10 - 40 CI, PB und DR werden auf die schon bekannten Werte festgelegt. Z wird als Zähler für die auftretenden Impulse verwendet und zu Anfang auf 0 gesetzt.

50 - 60 Hier werden alle Bits mit Ausnahme von PB0 als Ausgang programmiert und anschliesend mit dem Wert 0 geladen. Das hat den Vorteil, das wir die Bits 1 bis 7 nicht beachten brauchen. Diese Bits sind jetzt immer 0. Damit erhalten wir in Abhängigkeit von der Lichtschranke immer nur den Wert 0 oder 1.

70 - 80 Solange am PB0 ein Low anliegt, schleift das Programm in diesen beiden Zeilen. Erst wenn am Eingang ein High auftritt, gelangt das Programm in die Zeile

90 Hier wird der Zähler Z um eins erhöht.

100 Der aktuelle Zählerwert wird angezeigt. CHR$(147) ist die Anweisung den Bildschirm zu löschen. Damit erscheint der Wert immer in der linken oberen Ecke.

110 - 130 In den beiden ersten Zeilen schleift das Programm für die Dauer des High am Eingang. Erst wenn das Signal wieder nach Low wechselt, gelangt das Programm in die Zeile 130 und springt zur Zeile 70, wo auf eine erneute High-Flanke gewartet wird.

Mit diesem Programm lassen sich maximal 10 Impulse pro Sekunde zählen. Wenn auf eine andauernde Anzeige verzichtet werden kann, lassen sich allerdings wesentlich mehr Impulse pro Sekunde messen.

Statt dessen kann in der Zeile 100 des Programms beispielsweise eine Abfrage auf einen bestimmten Zählerstand erfolgen.

Eine mögliche Änderung wäre z.B.

```
100 IF Z/100 = INT(Z/100) THEN PRINT CHR$(147) ; Z
```

Eine andere Methode der Impulsmessung ist die Verwendung des -FLAG-Eingangs. Dazu wird der Ausgang der Lichtschranke mit dem -FLAG-Eingang der CIA am Kontakt B des User-Ports verbunden. Die CIA hat im ICR (Interrupt Control Register) das Bit 5 für den FLAG-Eingang reserviert. Sobald an diesem Eingang eine negative Flanke auftritt, in unserem Beispiel also kein Licht mehr auf den Phototransistor fällt, wird das Bit 5 im ICR gestzt. Wie bereits erwähnt wird der Inhalt des ICR automatisch mit dem Lesen dieses Registers gelöscht.

Aber nun das entsprechende Programm.

```
10 CI = 56576
20 IC = CI + 13
30 WE = PEEK (IC)
40 IF WE <> 16 THEN GOTO 30
50 Z = Z + 1
60 PRINT CHR$(147);Z
70 GOTO 30
```

Dieses Programm ist um einiges kürzer als das zuvor Beschriebene. Schneller ist es allerdings nicht. Auch in diesem Programm wird für das Löschen des Bildschirms und die Ausgabe viel Zeit benötigt, in der keine Impulse gezählt werden können.

4.4.2 Stopuhr mit Lichtschranke

Für Sportveranstaltungen, aber auch bei vielen anderen Gelegenheiten, z.B zum Messen von Geschwindigkeiten ist eine genaue Stopuhr nötig. Für die Realisation einer solchen Uhr gibt es mehrere Möglichkeiten. Wir werden zwei Möglichkeiten in diesem Kapitel besprechen. Dazu verwenden wir zwei identische lichtempfindliche Schaltungen nach Kapitel 4.4.
Die Ausgangsstufen werden gemeinsam an den -FLAG-Eingang gelegt.
Die beiden Lichtschranken werden so montiert, daß eine Lichtschranke am Start, die zweite am Ziel angebracht ist.
Bei einem Rundkurs wie z.B. einer Modellbahn kommen Sie natürlich auch mit einer Lichtschranke aus. Diese wird dann am Ziel montiert, das ja gleichermaßen Start und Ziel ist.
Tritt nun am -FLAG-Eingang ein negativer Spannungssprung auf, wird die Messung gestartet. Mit der zweiten negativen Flanke wird dann die Messung gestoppt und das Ergebnis angezeigt.
Im ersten Beispiel benutzen wir die eingebaute Uhr, die vom Betriebssystem unterstützt wird. Bei dieser Uhr wird die Zeit in der Variablen TI$ zur Verfügung gestellt.

```
 10 CI = 56576
 20 IC = CI + 13
 30 WE = PEEK (IC)
 40 PRINTCHR$(147) : PRINT"       ***  STOPUHR  *** "
 50 PRINT : PRINT"    DAS PROGRAMM IST AKTIVIERT"
 60 PRINT : PRINT"    UND  WARTET AUF  DEN START"
 70 WE = PEEK (IC)
 80 IF WE <> 16 THEN GOTO 70
 90 TI$ = "000000"
100 PRINTCHR$(147)
```

```
110 PRINT"    DIE MESSUNG IST GESTARTET"
120 PRINT : PRINT : PRINT
130 WE = PEEK (IC)
140 IF WE <> 16 THEN GOTO 130
150 A$ = TI$
160 PRINT"    DIE GEMESSENE ZEIT BETRAEGT :"
170 PRINT
180 HR = VAL(LEFT$(A$,2))
190 MI = VAL(MID$(A$,3,2))
200 SE = VAL(RIGHT$(A$,2))
210 PRINT"       ";HR;"STUNDEN"
220 PRINT"       ";MI;"MINUTEN"
230 PRINT"       ";SE;"SEKUNDEN"
240 PRINT:PRINT:PRINT"    DRUECKEN SIE  'F1' UM EINE"
250 PRINT:PRINT"    NEUE MESSUNG VORZUBEREITEN"
260 GETA$
270 IFA$=CHR$(133)THENRUN
280 GOTO260
```

Dieses Stopuhr ist natürlich nicht besonders genau. Die vom Betriebssystem unterstützte Uhr hat am Tag eine Abweichung von bis zu einer halben Stunde. Auch werden nur ganze Sekunden angezeigt.
Aber der C64 hat ja auch zwei sogenannte Echtzeituhren, die von der Netzfrequenz gesteuert werden. Diese Uhren sind für unseren Zweck sehr gut verwendbar. Zum einen haben sie durch die Netztriggerung eine hohe Ganggenauigkeit, zum anderen stellen sie auch Zehntel Sekunden zur Verfügung.
Die Uhren sind in den CIAs integriert und über die Register 8, 9, 10 und 11 zu erreichen. Zusätzlich werden von der Uhr noch die beiden höchstwertigen Bits in den Controlregistern CRA und CRB belegt.
Bit 7 im CRA bestimmt, ob die Uhr mit 50 Hz oder den in den USA gebräuchlichen 60 Hz Netzfrequenz getriggert werden

soll. Ein gesetztes Bit bedeutet 50 Hz.
Die Echtzeituhr hat außerdem die Möglichkeit, zu einer vorgewählten Zeit einen Alarm auszulösen. Um für Uhrzeit und Alarmzeit nicht jeweils 4 Register zu verwenden, wird mit dem Bit 7 des CRB zwischen den beiden Möglichkeiten hin- und hergeschaltet. Ist das Bit gesetzt, so kann man die Alarmzeit stellen, bei gelöschtem Bit die Uhrzeit.
Die Uhrzeit steht in den Registern im BCD-Format. Damit ist eine relativ einfache Berechnung nötig, um die Uhrzeit auf dem Bildschirm anzuzeigen.
Besondere Bedeutung haben TODH und TODT, also Stunden- und 1/10-Sekunden-Register. Wird das Stundenregister gelesen, so wird die aktuelle Uhrzeit in einen Zwischenspeicher geschrieben, aus dem nun die Uhrzeit abgerufen werden kann. Die Uhr läuft jetzt zwar weiter, eventuelle Überträge stören aber nicht, da das Zwischenregister erst wieder freigegeben wird, wenn das 1/10-Sek.-Register gelesen wird. Dadurch kann man sich nach dem Lesen von TODH beliebig Zeit lassen, es wird immer der Zeitpunkt angezeigt, zu dem TODH gelesen wurde.
Wird die Uhrzeit gestellt, dann sollte man das Stundenregister zuerst laden. Dadurch wird die Uhr angehalten. Danach können die Register TODS und TODM beschrieben werden. Mit dem Laden von TODT startet dann die Uhr mit der gewünschten Uhrzeit.
Um die Genauigkeit der Uhren auszunutzen, ist eine Programmierung in Maschinensprache nötig. Das nachfolgend abgedruckte Assemblerprogramm stellt eine komfortable Bedienung der zwei CIAs dar. Dabei wird CIA1 benutzt, um die aktuelle Uhrzeit anzuzeigen, CIA2 wird als eigentliche Stopuhr benutzt. Auch diese Stopuhr funktioniert wieder mit Steuerung über den -FLAG-Eingang. In diesem Falle wird aber mit Auftreten eines Impulses ein nicht maskierbarer Interrupt (NMI) ausgelöst, der dann sofort die Stopuhr

anhält.
Auch die Anzeige der Uhr- und Stopzeit geschieht im Interrupt. Dieser Interrupt wird aber nicht vom Programm erzeugt. Wir verwenden den normalen Systeminterrupt, auch Timerinterrupt genannt, der ein maskierbarer Interrupt ist (IRQ). Dieser wird ca. alle 16 Millisekunden automatisch ausgelöst.
Der Timerinterrupt fragt die Tastatur ab, erhöht die Uhrzeit im TI$ (der Systemuhr aus dem vorigen Beispiel), lässt den Cursor blinken und erledigt verschiedene andere Dinge mehr.
Da die Zeiger auf die IRQ- und NMI-Routinen im Ram liegen, können wir sie einfach auf unsere Routinen umlenken, und danach auf die alten Systemroutinen springen.
Aber betrachten wir zuerst einmal das Assemblerprogramm.

```
CEB5                *=   $CEB5
CEB5        FARBE   =    $0286      ; momentane Zeichenfarbe
CEB5        BUSY1   =    $FB        ; Flag für Uhrzeitanzeige
CEB5        BUSY2   =    $FC        ; Flag für Stopuhranzeige
CEB5        COLRAM  =    $F9
CEB5        VIDEO   =    $F7
CEB5        CIA1    =    $DC00
CEB5        CIA2    =    $DD00
CEB5        NMIALT  =    $FE47
CEB5        NMIVEC  =    $0318
CEB5        IRQALT  =    $EA31
CEB5        IRQVEC  =    $0314
CEB5        TODT1   =    CIA1+8     ; Zehntelsekundenreg.
CEB5        TODS1   =    CIA1+9     ; Sekundenregister
CEB5        TODM1   =    CIA1+10    ; Minutenregister
CEB5        TODH1   =    CIA1+11    ; Stundenregister
CEB5        TODT2   =    CIA2+8
```

```
CEB5          TODS2    =    CIA2+9      ; wie
CEB5          TODM2    =    CIA2+10     ; CIA 1
CEB5          TODH2    =    CIA2+11
CEB5          ICR1     =    CIA1+13     ; Interruptcontolreg.
CEB5          ICR2     =    CIA2+13
CEB5          CRA1     =    CIA1+14
CEB5          CRB1     =    CIA1+15     ; Controlregister
CEB5          CRA2     =    CIA2+14     ; CIA 1 & 2
CEB5          CRB2     =    CIA2+15
CEB5          DISP     =    $04DA       ; Videoram für Anzeige
CEB5          COL      =    $D8DA       ; entsp. Farbram
              ;========== INTERRUPT INIT ==========
CEB5 78             SEI                 ;
CEB6 A9 00          LDA  #0             ; Stopzeitanzeige
CEB8 85 FC          STA  BUSY2          ; ausschalten
CEBA A9 14          LDA  #<IRQNEU       ; Interuptvector
CEBC 8D 14 03       STA  IRQVEC         ; auf unser Prg.
CEBF A9 CF          LDA  #>IRQNEU       ; verändern
CEC1 8D 15 03       STA  IRQVEC+1
CEC4 A9 DA          LDA  #<COL
CEC6 85 F9          STA  COLRAM         ; Farbram und
CEC8 A9 D8          LDA  #>COL          ; Videoramadressen
CECA 85 FA          STA  COLRAM+1       ; in den
CECC A9 DA          LDA  #<DISP         ; Pointern ablegen
CECE 85 F7          STA  VIDEO
CED0 A9 04          LDA  #>DISP
CED2 85 F8          STA  VIDEO+1
CED4 AD 0D DD       LDA  ICR2           ; ICR2 löschen
CED7 A9 AC          LDA  #<NMINEU
CED9 8D 18 03       STA  NMIVEC         ; NMI-Vector
CEDC A9 CF          LDA  #>NMINEU       ; verbiegen
CEDE 8D 19 03       STA  NMIVEC+1
CEE1 AD 0E DC       LDA  CRA1           ; 50 Hz Trigger
CEE4 09 80          ORA  #$80           ; einschalten
```

```
CEE6 8D 0E DC          STA  CRA1
CEE9 AD 0F DC          LDA  CRB1       ; Uhrzeit wählen,
CEEC 29 7F             AND  #$7F       ; Alarmzeit ausschalten
CEEE 8D 0F DC          STA  CRB1
CEF1 AD 0E DD          LDA  CRA2       ; und das selbe
CEF4 09 80             ORA  #$80       ; bei CIA 2
CEF6 8D 0E DD          STA  CRA2
CEF9 AD 0F DD          LDA  CRB2
CEFC 29 7F             AND  #$7F
CEFE 8D 0F DD          STA  CRB2
CF01 A9 00             LDA  #0
CF03 8D 08 DD          STA  TODT2      ; 1/10 Sec. auf 0
CF06 8D 0B DD          STA  TODH2      ; Stunden- und die anderen
CF09 8D 0A DD          STA  TODM2      ; Register auf 0, Uhr
CF0C 8D 09 DD          STA  TODS2      ; anhalten
CF0F 8D 08 DC          STA  TODT1      ; Uhr 1 starten
CF12 58                CLI
CF13 60                RTS
                ;
                ;======= IRQ NEU ============
                ;
CF14 78         IRQNEU SEI
CF15 A0 09             LDY  #9
CF17 AD 86 02          LDA  FARBE      ; Farbram füllen
CF1A 91 F9     LOOP1   STA  (COLRAM),Y
CF1C 88                DEY
CF1D 10 FB             BPL  LOOP1
CF1F A0 5A             LDY  #90
CF21 91 F9     LOOP2   STA  (COLRAM),Y
CF23 88                DEY
CF24 C0 4F             CPY  #79
CF26 D0 F9             BNE  LOOP2
CF28 A9 7F             LDA  #$7F       ; NMI vom Flageingang
CF2A 8D 0D DD          STA  ICR2       ; zulassen.
```

```
CF2D A9 90              LDA  #%10010000
CF2F 8D 0D DD           STA  ICR2
                ;
                ;========= CIA 1 UHRZEIT =======
                ;
CF32 A5 FB              LDA  BUSY1      ; Uhrzeit anzeigen?
CF34 F0 2A              BEQ  ENDE1
CF36 A0 00              LDY  #0         ; Positionszähler
CF38 AD 0B DC           LDA  TODH1      ; Stundenregister
CF3B 10 07              BPL  WEI        ; AM oder PM?
CF3D 29 7F              AND  #$7F
CF3F 18                 CLC             ; bei PM
CF40 F8                 SED             ; 12 addieren
CF41 69 12              ADC  #$12       ; für 0 bis 23
CF43 D8                 CLD
CF44 20 99 CF WEI       JSR  CRUNCH     ; Ergebnis ausgeben
CF47 A9 3A              LDA  #":"
CF49 91 F7              STA  (VIDEO),Y  ; ':' ausgeben und
CF4B C8                 INY             ; Position erhöhen
CF4C AD 0A DC           LDA  TODM1      ; Minutenregister
CF4F 20 99 CF           JSR  CRUNCH     ; ausgeben
CF52 A9 3A              LDA  #":"       ; und wieder
CF54 91 F7              STA  (VIDEO),Y  ; ':'
CF56 C8                 INY
CF57 AD 09 DC           LDA  TODS1      ; Sekunden anzeigen
CF5A 20 99 CF           JSR  CRUNCH
CF5D AD 08 DC           LDA  TODT1      ; Register wieder freigeben
                ;
                ; ======= CIA 2 STOPZEIT ========
                ;
CF60 A5 FC    ENDE1     LDA  BUSY2      ; Stopzeit anzeigen ?
CF62 F0 32              BEQ  ENDE2
CF64 A0 50              LDY  #80        ; Position für Stopzeit
CF66 AD 0B DD           LDA  TODH2
```

```
CF69 10 07            BPL  WEI1       ; wie bei CIA 1
CF6B 29 7F            AND  #$7F
CF6D 18               CLC
CF6E F8               SED
CF6F 69 12            ADC  #$12
CF71 D8               CLD
CF72 20 99 CF WEI1    JSR  CRUNCH
CF75 A9 3A            LDA  #":"
CF77 91 F7            STA  (VIDEO),Y
CF79 C8               INY
CF7A AD 0A DD         LDA  TODM2
CF7D 20 99 CF         JSR  CRUNCH
CF80 A9 3A            LDA  #":"
CF82 91 F7            STA  (VIDEO),Y
CF84 C8               INY
CF85 AD 09 DD         LDA  TODS2
CF88 20 99 CF         JSR  CRUNCH
CF8B A9 3A            LDA  #":"
CF8D 91 F7            STA  (VIDEO),Y
CF8F C8               INY
CF90 AD 08 DD         LDA  TODT2      ; 1/10 Sek lesen
CF93 20 A3 CF         JSR  H1         ; und anzeigen
CF96 4C 31 EA ENDE2   JMP  IRQALT     ; zur alten IRQ Routine
                ;
                ;======== CRUNCH ===========
                ;
CF99 48       CRUNCH  PHA             ; wird noch benötigt
CF9A 4A               LSR
CF9B 4A               LSR             ; 10er Stelle in
CF9C 4A               LSR             ; Lownibble übertragen
CF9D 4A               LSR
CF9E 09 30            ORA  #"0"       ; Zahl bilden
CFA0 91 F7            STA  (VIDEO),Y  ; und anzeigen
CFA2 68               PLA             ; alter Wert zurück
```

```
CFA3 C8       H1        INY             ; Position erhöhen
CFA4 29 0F              AND  #$0F       ; 10er Stelle ausblenden,
CFA6 09 30              ORA  #"0"       ; Zahl bilden und
CFA8 91 F7              STA  (VIDEO),Y  ; anzeigen
CFAA C8                 INY             ; Position erhöhen und
CFAB 60                 RTS             ; zurück
                ;
                ;======= NMI NEU ===========
                ;
CFAC 48       NMINEU    PHA
CFAD 8A                 TXA             ; alle Register retten
CFAE 48                 PHA
CFAF 98                 TYA
CFB0 48                 PHA
CFB1 A5 FC              LDA  BUSY2      ; läuft Stopuhr ?
CFB3 D0 13              BNE  ZIEL       ; wenn ja dann zum Ziel
CFB5 A9 00    START     LDA  #$0        ; Stopuhr bei 0
CFB7 8D 08 DD           STA  TODT2      ; starten
CFBA A0 00              LDY  #0         ; weitere NMIs
CFBC AD 0D DD PRELL     LDA  ICR2       ; verhindern und
CFBF C8                 INY             ; evtl. Prellen
CFC0 D0 FA              BNE  PRELL      ; beseitigen
CFC2 A9 01              LDA  #$1        ; Anzeige
CFC4 85 FC              STA  BUSY2      ; einschalten
CFC6 D0 1B              BNE  RET
CFC8 AD 0B DD ZIEL      LDA  TODH2      ; Uhr anhalten
CFCB A0 00              LDY  #0         ; weitere NMIs
CFCD AD 0D DD PREL1     LDA  ICR2       ; verhindern und
CFD0 C8                 INY             ; evtl. Prellen
CFD1 D0 FA              BNE  PREL1      ; beseitigen
CFD3 A9 00              LDA  #0
CFD5 85 FC              STA  BUSY2
CFD7 8D 08 DD           STA  TODT2      ; Uhr auf 00:00:00
CFDA 8D 0B DD           STA  TODH2      ; setzen
```

```
CFDD 8D 0A DD          STA  TODM2
CFE0 8D 09 DD          STA  TODS2
CFE3 68       RET      PLA
CFE4 A8                TAY            ; Register zurück
CFE5 68                PLA            ; holen und
CFE6 AA                TAX            ; NMI beenden
CFE7 68                PLA
CFE8 40                RTI
                ;
                ;========= ENDE ===============
                ;
CFE9 78                SEI
CFEA A9 31             LDA  #<IRQALT
CFEC 8D 14 03          STA  IRQVEC    ; NMI und IRQ
CFEF A9 EA             LDA  #>IRQALT  ; auf die ursprüng-
CFF1 8D 15 03          STA  IRQVEC+1  ; lichen Werte zurück
CFF4 A9 47             LDA  #<NMIALT  ; schreiben und
CFF6 8D 18 03          STA  NMIVEC    ; zurück
CFF9 A9 FE             LDA  #>NMIALT
CFFB 8D 19 03          STA  NMIVEC+1
CFFE 58                CLI
CFFF 60                RTS
```

Lesern mit Kenntnissen der Maschinenspracheprogrammierung sollte dies Programm eigentlich keine Schwierigkeiten bereiten. Für die 'NUR-BASIC'-Programmierer stellen dies und die noch folgenden Programme vielleicht eine Herausforderung dar, sich doch einmal mit dieser Art der Programmierung anzufreunden.
Wir wollen nicht das ganze Programm im Detail besprechen, sondern nur auf einige besondere Punkte und den Gebrauch des Programms zu sprechen kommen.
Das Programm hat eine Genauigkeit von ca. 100 Millisekunden.

Dieser Fehler liegt am Aufbau der CIA. Da die Netzfrequenz asynchron zur Taktfrequenz des Rechners läuft, kann man keine Aussage über die Länge der ersten '1/10-Sekunde' machen. Diese erste gemessene Zeiteinheit wird in der Regel kürzer als eine richtige 1/10-Sekunde sein. Dieser Fehler ist aber mit programmtechnischen Maßnahmen nicht zu verhindern.
Um die Eingabe des Programms zu erleichtern, sei hier noch ein Basic-Laderprogramm abgedruckt.

```
100 FOR I = 52917 TO 53247
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 120,169,  0,133,252,169, 20,141, 20,  3,169,207
130 DATA 141, 21,  3,169,218,133,249,169,216,133,250,169
140 DATA 218,133,247,169,  4,133,248,173, 13,221,169,172
150 DATA 141, 24,  3,169,207,141, 25,  3,173, 14,220,  9
160 DATA 128,141, 14,220,173, 15,220, 41,127,141, 15,220
170 DATA 173, 14,221,  9,128,141, 14,221,173, 15,221, 41
180 DATA 127,141, 15,221,169,  0,141,  8,221,141, 11,221
190 DATA 141, 10,221,141,  9,221,141,  8,220, 88, 96,120
200 DATA 160,  9,173,134,  2,145,249,136, 16,251,160, 90
210 DATA 145,249,136,192, 79,208,249,169,127,141, 13,221
220 DATA 169,144,141, 13,221,165,251,240, 42,160,  0,173
230 DATA  11,220, 16,  7, 41,127, 24,248,105, 18,216, 32
240 DATA 153,207,169, 58,145,247,200,173, 10,220, 32,153
250 DATA 207,169, 58,145,247,200,173,  9,220, 32,153,207
260 DATA 173,  8,220,165,252,240, 50,160, 80,173, 11,221
270 DATA  16,  7, 41,127, 24,248,105, 18,216, 32,153,207
280 DATA 169, 58,145,247,200,173, 10,221, 32,153,207,169
290 DATA  58,145,247,200,173,  9,221, 32,153,207,169, 58
300 DATA 145,247,200,173,  8,221, 32,163,207, 76, 49,234
310 DATA  72, 74, 74, 74, 74,  9, 48,145,247,104,200, 41
320 DATA  15,  9, 48,145,247,200, 96, 72,138, 72,152, 72
330 DATA 165,252,208, 19,169,  0,141,  8,221,160,  0,173
```

```
340 DATA  13,221,200,208,250,169,  1,133,252,208, 27,173
350 DATA  11,221,160,  0,173, 13,221,200,208,250,169,  0
360 DATA 133,252,141,  8,221,141, 11,221,141, 10,221,141
370 DATA   9,221,104,168,104,170,104, 64,120,169, 49,141
380 DATA  20,  3,169,234,141, 21,  3,169, 71,141, 24,  3
390 DATA 169,254,141, 25,  3, 88, 96
400 IF S <> 42208 THEN PRINT "FEHLER IN DATAS !!" : END
410 PRINT "OK"
```

Von Basic aus kann das Programm mit SYS 52917 aktiviert werden. Da es vollständig interruptgesteuert ist, kann nun jedes gewünschte Basicprogramm gestartet werden. Der SYS kann sogar im Direktmodus eingegeben werden.
Besondere Bedeutung haben die Speicherzellen 251 und 252 in der Zeropage. Nach dem SYS werden die Uhrzeit und die Stopzeit sofort angezeigt. Werden nun in die Speicherzellen 251 und 252 eine 0 gePOKEd, dann laufen Uhrzeit und Stopzeit (wenn aktiviert) zwar weiter, die Anzeige wird aber ausgeschaltet. Durch eine 1 in die Speicherzellen kann die Anzeige wieder eingeschaltet werden. Die Speicherzelle 252 ist für die Anzeige der Stopzeit zuständig, 251 für die Anzeige der Uhrzeit.
Start und Stop der Stopzeit geschieht wie schon erwähnt mit einem NMI. Da keine Abfrage über die Quelle des Interrupts vorgenommen wird, kann die Stopuhr auch mit der RESTORE-Taste ein- und ausgeschaltet werden.
Bedenken Sie diesen Punkt, wenn Sie mit dem Programm experimentieren. Solange es aktiviert ist, kann RUN/STOP-RESTORE nicht verwendet werden, um aus einem laufenden Programm herauszukommen. Aus diesem Grund kann durch SYS 53225 das Programm deaktiviert werden. Dadurch werden die alten Interruptroutinen wieder eingeschaltet, auch RUN/STOP-RESTORE hat danach wieder seine ursprüngliche Funktion.

Das nachfolgende Basicprogramm setzt voraus, daß das Maschinenspracheprogramm bereits im Speicher steht. Dazu können Sie den Basicloader benutzen. Es gestattet das Setzen der Uhrzeit und zeigt die Verwendung der verschiedenen Möglichkeiten. Betrachten Sie dies Programm nicht als fertiges, endgültiges Programm. Es soll nur die Möglichkeiten aufzeigen und Sie zum Experimentieren anregen.

Betrachten wir kurz das Basicprogramm.

```
 10 C = 56328
 20 SYS 52917
 30 SYS 53225
 40 PRINT CHR$(147)
 50 PRINT" BITTE GEBEN SIE DIE UHRZEIT "
 60 PRINT
 70 PRINT"    IM FORMAT HHMMSS EIN"
 80 PRINT
 90 PRINT"            ";:   INPUT A$
100 IF LEN(A$) <> 6 THEN 50
110 H = VAL(LEFT$(A$,2))
120 M = VAL(MID$(A$,3,2))
130 S = VAL(RIGHT$(A$,2))
140 IF H > 23 THEN 50
150 IF H > 11 THEN H = H + 68
160 POKE C+3 , 16*INT(H/10)+H-INT(H/10)*10
170 IF M > 59 THEN 50
180 POKE C+2 , 16*INT(M/10)+M-INT(M/10)*10
190 IF S > 59 THEN 50
200 POKE C+1 , 16*INT(S/10)+S-INT(S/10)*10
210 POKE C , 0
220 PRINT CHR$(147)
230 SYS 52917 : POKE 252,1 : POKE 251,1
240 PRINT"        S T O P U H R"
```

```
250 PRINT : PRINT : PRINT : PRINT"     UHRZEIT"
260 PRINT : PRINT"     STOPZEIT"
270 FOR I = 1TO5
280 PRINT CHR$(17);
290 NEXT
300 PRINT"     F1 = STOPUHR RUECKSETZEN"
310 PRINT
320 PRINT"     F3 = ZWISCHENZEIT ANZEIGEN"
330 PRINT
340 PRINT"     F7 = PROGRAMM BEENDEN"
350 GET AN$ : IF AN$ = "" THEN 350
360 IF AN$ = CHR$(133) THEN 400
370 IF AN$ = CHR$(134) THEN 420
380 IF AN$ = CHR$(136) THEN 440
390 GOTO 350
400 POKE 252,1 : FOR I = 1TO100 : NEXT : POKE 252,0
410 GOTO 350
420 POKE 252,0 : FOR I = 1TO2000 : NEXT : POKE252,1
430 GOTO 350
440 PRINT CHR$(19) : FOR I = 1TO20 : PRINTCHR$(17); :NEXT
450 PRINT"       WIRKLICH ENDE";
460 INPUT EN$
470 IF EN$ = "J" THEN 510
480 PRINT CHR$(19) : FOR I = 1TO20 : PRINTCHR$(17); :NEXT
490 PRINT"                                          "
500 GOTO 350
510 SYS 53225
520 PRINT CHR$(147)
530 PRINT : PRINT : PRINT
540 PRINT"          STOPUHR ENDE !"
```

Nach dem Eingeben der Uhrzeit erscheint auf dem Bildschirm die Uhrzeit und die Stopuhr wird angezeigt. Die Stopuhr steht auf 00:00:00: 0. Jetzt kann durch einen NMI, also ein

Signal am -FLAG-Eingang oder Drücken der RESTORE-Taste die Stopuhr gestartet werden. wird jetzt F3 gedrückt, so wird für ca. 2 Sekunden die Anzeige der Stopuhr angehalten und ein Zwischenwert angezeigt. Danach läuft die Stopuhr weiter. Kommt jetzt ein zweiter NMI, so wird die Stopuhr ganz angehalten und die gemessene Zeit wird angezeigt. Wird jetzt Fl gedrückt, so verschwindet die gemessene Zeit, die Stopuhr steht wieder auf 00:00:00: 0. Jetzt kann eine neue Messung vorgenommen werden.
Wird nach dem Ziel-Impuls die Fl-Taste nicht gedrückt, so beginnt die Stopuhr automatisch wieder bei 0. Das Rücksetzen ist also nicht erforderlich um einen neuen Meßvorgang einzuleiten.

4.5 Die Verwendung der Timer

In jeder CIA sind zwei 16-Bit-Timer integriert. Jeder dieser Timer verfügt über mehrere Betriebsarten. So können sehr einfach Zeitschleifen, Impulsketten oder Rechteckspannungen erzeugt werden. Aber auch externe Signale können über den CNT-Pin gezählt werden.
Die Betriebsart wird in den Controlregistern CRA und CRB bestimmt. Die genaue Programmierung der Betriebsarten können Sie dem Kapitel 4 entnehmen.

Wir wollen uns sofort an die Arbeit begeben und die CIAs für einen universellen Rechteckgenerator im NF-Bereich (bis ca. 20 KHz) verwenden.

```
100 C2 = 56576
110 AL = C2+4
120 AH = C2+5
130 CA = C2+14
140 TF = 985248
150 TH = TF/2
160 POKE CA , 23
170 PRINT CHR$(147)
180 PRINT : PRINT : PRINT
190 PRINT"    FREQUENZ IN HERTZ EINGEBEN"
200 PRINT : INPUT"          ";FR
210 FW = TH/FR
220 HB = FW/256
230 HB = INT(HB)
240 LB = FW-HB*256
250 LB = INT(LB+.5)
260 IF LB = 256 THEN LB = 0 : HB = HB+1
270 POKE AH,HB : POKE AL,LB-1
280 PRINT : PRINT : PRINT : PRINT
```

```
290 PRINT"   DIE ERZEUGTE FREQUENZ BETRAEGT"
300 PRINT : PRINT"                                          "
310 PRINT CHR$(145); : PRINT"           ";
320 PRINT (INT(TH/(HB*256+LB)*10))/10;"HERTZ"
330 PRINT : PRINT : PRINT : PRINT
340 PRINT"     F1 FUER FREQUENZAENDERUNG"
350 PRINT"     F7 FUER PROGRAMMENDE"
360 GET A$ : IF A$ = "" THEN 360
370 IF A$ = CHR$(133) THEN PRINT CHR$(19) : GOTO 180
380 IF A$ = CHR$(136) THEN 400
390 GOTO 360
400 PRINT CHR$(147)
410 PRINT : PRINT : PRINT"         ENDE DES PROGRAMMS!"
420 PRINT : PRINT"    DIE FREQUENZ AN PB6 BLEIBT NUN "
430 PRINT : PRINT"    BIS ZU EINER UMPROGRAMMIERUNG "
440 PRINT : PRINT"    VON TIMER A DER CIA 2 ERHALTEN."
450 PRINT : PRINT : PRINT"         ACHTUNG !"
460 PRINT : PRINT"    AUCH RUN STOP/RESTORE LOESCHT"
470 PRINT : PRINT"    DIE TIMER IN DEN CIAS."
```

Dieses Programm verwendet den Timer A der CIA 2.
Durch die Programmierung erzeugt der Timer Rechteck-Signale am User-Port Pin PB6. Zur Einstellung der Frequenz ist aber nicht wie sonst üblich ein Poti oder sonstiges Abstimmbauteil nötig, sie wird einfach von der Tastatur aus eingegeben.
Die Frequenzen können sich in Bereich von 8 Hertz bis zu einem viertel der Taktfrequenz, also 246312 Hertz bewegen.
Die erzeugte Frequenz wird von der Taktfrequenz des C64 abgeleitet. Diese ist quarzstabilisiert und somit ist auch die Rechteckspannung quarzstabil.
Durch das verwendete Prinzip der Signalerzeugung durch Frequenzteilung lassen sich die gewünschten Frèquenzen nicht ganz genau erreichen. Je höher die gewünschte Frequenz ist,

desdo größer werden die erzielbaren Frequenzschritte. So beträgt die höchste Frequenz 246 KHz, die nächst kleiner Frequenz beträgt aber schon 164 KHZ.
Im Frequenzbereich von 10 Hz bis 16 KHz ist der Fehler aber immer kleiner als .5%, in den meisten Fällen sogar kleiner als 0.1%, also kleiner als 1 Promille. Diese Einstellgenauigkeit ist bei vielen einstellbaren Rechteckoszillatoren nur schwer zu erreichen.
Außerdem haben herkömmliche Oszillatorschaltungen fast immer mit dem Problem der Temparaturdrift zu kämpfen. Diese Drift ist bei einem Quarzoszillator mit fester Frequenz schon von Hause aus sehr gering. Da die Quarzfrequenz aber noch geteilt wird , minimiert jede zusätzliche Teilung auch den durch Drift hervorgerufenen Fehler.

Aber kommen wir zu der Funktion des Programms.

100 - 150 Definition der verwendeten Variablen. Die Variable TF enthält die Taktfrequenz des C64 in Hertz, TH die halbe Taktfrequenz.

160 Timer A der CIA 2 wird vorbereitet. Die Betriebsart im einzelnen :
Timer Start, Unterläufe werden an PB 6 signalisiert wobei jeder Unterlauf das Portbit in die jeweils andere Polarität schaltet, Continuous-Mode, der Zähler startet nach dem Ablauf selbsttätig, der Zähler kann zu jedem Zeitpunkt mit neuen Werten geladen werden, als Taktquelle wird der Systemtakt verwendet

170 - 200 Die gewünschte Frequenz wird abgefragt.

210 TH geteilt durch die Soll - Frequenz ergibt die Anzahl der zu zählenden Impulse für den Timer.

220 - 230 Aus diesem Wert wird das Highbyte HB für den

Timer gewonnen.

240 - 250 Das Lowbyte des Timerwertes wird gewonnen und gerundet. Durch die Rundung wird immer der nächstliegende Frequenzwert gewählt.

260 Sollte durch die Rundung LB größer als 255 werden, so wird ein Übertrag in das HB gebracht und das LB gelöscht.

270 Die errechneten Werte werden in den Timer gePOKEd. Wichtig ist hierbei, das der Wert für das LB um 1 erniedrigt werden muss. Diese Reduzierung ist in der Konstruktion der CIAs begründet.

280 - 320 Zur Kontrolle wird in der Zeile 320 aus den errechneten Werten auf die tatsächliche Frequenz zurückgerechnet und diese dann auf eine Nachkommastelle gerundet angezeigt.

330 - 470 Diese Zeilen stellen ein kleines Menü dar, in dem gewählt werden kann zwischen Änderung der Frequenz und beenden des Programms. Wird das Programm beendet, so erscheint eine Meldung aus den Zeilen 400 - 470, ansonsten kann eine neue Frequenz eingegeben werden.

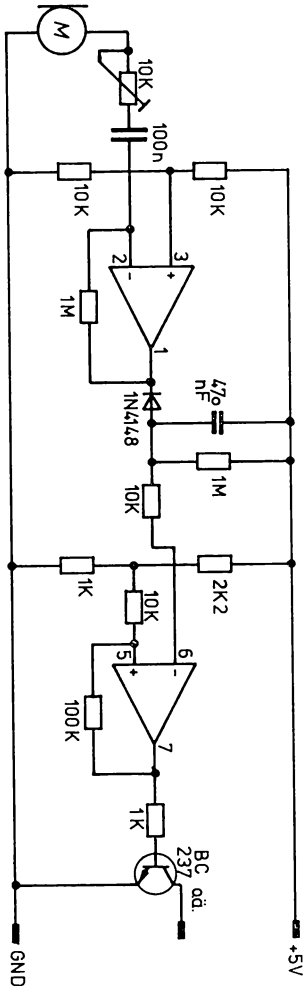
Noch ein Wort zu der Zeile 270.
Wird der Timer in der zuvor besprochenen Betriebsart betrieben, so ist das kleinste einstellbare Teilerverhältnis 1/4. Diese Verhältnis erhält man durch Laden des Highbytes des Zählers mit 0, während das Lowbyte mit 1 geladen wird. Werden beide Timerregister mit 0 geladen, so bleibt der Ausgang PB6 (PB7 für Timer B) auf High.
Wenn wir nun einmal die Werte für eine Frequenz von 248 KHz nach der im Programm enthaltenen Formel berechnen, so erhalten wir nach Zeile 210 für FW einen Wert von 1.986. Damit ist HB = 0. Das Lowbyte LB wird nach der Rundung zu 2.

Damit geladen würde der Timer aber eine Frequenz von ca 164 KHz erzeugen. Wird aber LB um eins vermindert, so stimmt die erwünschte Frequenz mit der Erzeugten überein.

Wenn mit den Rechtecksignalen nur TTL-Schaltungen versorgt werden, so erübrigt sich in der Regel eine zusätzliche Ausgangsstufe. In diesem Falle darf aber die Last am Ausgang nicht zu groß werden (ca. lmA). Außerdem reagiert die CIA gegen Überspannung in der Regel durch Totalausfall. Darum können vorsichtige Leser eine kleine Ausgangsstufe vorsehen, die aus nur einem IC 74128 besteht und sich leicht auf einer Lochrasterplatine an einem User-Port-Stecker montieren lässt und den Rechner wirkungsvoll schützt.
Dies IC enthält vier identische NOR-Gatter mit zwei Eingängen. Das Besondere dieses ICs ist die Tatsache, daß es speziell für Anwendungen mit erhöhtem Ausgangsstrom, z.B. Leitungstreiberschaltungen, entwickelt wurde. Jedes Gatter vermag einen Strom von 48mA liefern. Das sollte in allen Digitalschaltungen ausreichen.
Durch einen 47 Ohm-Widerstand in der Ausgangsleitung ist die Stufe auf jeden Fall sowohl gegen Plus als auch Masse kurzschlußsicher.
Da im 74128 vier gleiche Stufen integriert sind, können beide Timer der CIA 2 gepuffert werden. Wenn Sie nun noch das abgedruckte Programm erweitern, erhalten Sie zwei unabhängig voneinander einstellbare Rechteckgeneratoren mit exzellenten Eigenschaften. Diese Erweiterung dürfte Ihnen an Hand der Programmbeschreibung recht leicht fallen.

4.6 Klatschschalter

In diesem Kapitel wollen wir uns einen einfachen Klatschschalter konstruieren.
Ein Klatschschalter ist eine Art Fernbedienung. Ein Mikrofon nimmt die Umgebungsgeräusche auf, die verstärkt und einer Schaltstufe zugeführt werden. Ab einer bestimmten Lautstärke wird ein Ausgangssignal erzeugt. Dieses Ausgangsignal kann dann zum Schalten verwendet werden.
In unserem Beispiel werden der Verstärker und die Schaltstufe mit einem Operationsverstärker vom Typ LM 324 aufgebaut. Wir verwenden zwei der vier integrierten Operationsverstärker.
Als Mikrofon sollte ein dynamisches Tauchspulenmikrofon mit einer Impedanz von ca. 200 Ohm verwendet werden. Damit lässt sich jede handelsübliche preiswerte Mikrofonkapsel verwenden. An den Frequenzgang des Mikrofons werden keine Anforderungen gestellt.
Der erste OP ist als einfacher Wechselspannungsverstärker geschaltet. Die Verstärkung dieser Stufe ist mit dem Poti einstellbar und liegt zwischen 200 und 1000-facher Verstärkung. Damit kann die Schaltung an unterschiedliche Mikrofone und Empfindlichkeiten angepasst werden.
Auf den Verstärker folgt eine Diodenstufe, die das verstärkte Mikrofonsignal gleichrichtet und einige Zeit speichert.
Dieses Gleichspannungssignal wird auf einen Schwellwertschalter geführt, der mit OP 2 aufgebaut ist. Damit erhalten wir am Ausgang ein sauber begrenztes Rechtecksignal, das einen Transistor ansteuert.
Dieser Transistor stellt die eigentliche Ausgangsstufe dar.



Jetzt stellt sich nur noch die Frage, wie wir den Klatschschalter möglichst einfach abfragen können. Dafür ist der Control-Port 1 ausgezeichnet geeignet.

Wenn Sie Besitzer eines Joysticks sind, werden Sie vielleicht festgestellt haben, daß das Betätigen des Joysticks Zeichen auf dem Bildschirm erscheinen läßt, wenn der Computer im Direkt-Modus ist. Diese Zeichen lassen sich in einem kleinen Basic-Programm mit einer GET-Schleife abfragen. Statt des Joysticks lassen sich natürlich auch Transistoren als Schalter einsetzten. Von dieser Möglichkeit machen wir beim Klatschschalter Gebrauch.

Wir wollen Pin 2 von Control-Port 1 als Eingang verwenden. Wird dieser Anschluß nach Masse gelegt, so wird ein CHR$ (95) in den Tastaturpuffer geschrieben. Das entspricht dem Pfeil nach links von der Tastatur.

Das folgende kleine Beispielprogramm ist keine besonders sinnvolle Anwendung des Klatschschalters, zeigt aber, wie eine mögliche Realisation

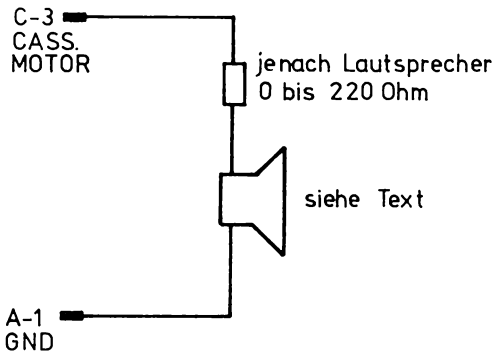
mit dieser Art der Abfrage aussehen kann. Mit etwas Phantasie lassen sich leicht andere Anwendungen programmieren.

```
100 BC = 53280
110 BG = 53281
120 PRINT CHR$(147)
130 PRINT"      KLATSCHSCHALTER"
135 PRINT
140 GET A$ : IF A$ = "" THEN 140
150 IF A$ <> CHR$(95) THEN 140
160 PRINT"         AKTIVIERT"
165 PRINT : PRINT
170 POKE BG , 1
180 TI$ = "000000"
190 GET A$
200 IF A$ = CHR$(95) THEN TI$="000000": Z=Z+1: POKE BC,Z
210 PRINTCHR$(145);"           "TI$
220 IFTI$<>"000005"THEN190
230 POKE BG,6
240 GOTO 120
```

Die Variablen BG und BC enthalten die Adressen für Bilschirmfarbe und Rahmenfarbe (Background und Border Colour). Wird ein erstes Mal geklatscht, so wechselt die Bildschirmfarbe auf Weiß. Gleichzeitig wird angezeigt, das der Klatschschalter aktiviert ist. Zusätzlich wird eine Uhr angezeigt. Wird nun innerhalb von 5 Sekunden ein weiteres Mal geklatscht, so wechselt die Rahmenfarbe und die Uhr wird auf 0 gesetzt. Jetzt kann wieder innerhalb von 5 Sekunden durch Klatschen die Rahmenfarbe geändert werden. Wird jedoch in den 5 Sekunden nicht geklatscht, so wechselt die Hintergrundfarbe wieder nach Blau und das Programm wartet auf eine neue Aktivierung.

4.7 Lautsprecher am Cassetten-Port

Die in diesem Kapitel beschriebene Möglichkeit der Tonerzeugung wird Ihnen auf den ersten Blick möglicherweise überflüssig erscheinen. Verfügt der C64 doch über geradezu enorme Möglichkeiten mit dem eingebauten SID Töne, Klänge und Melodien zu erzeugen. Allerdings ist dazu immer ein Fernsehgerät oder ein Verstärker notwendig. Für viele Aufgaben, wie zum Beispiel Überwachung von Räumen, Temparaturen oder Produktionsabläufen ist ein andauernder Betrieb eines Fernsehers garnicht notwendig und natürlich auch nicht besonders Energiesparend.
Der Computer kann zum Beispiel in den Nachtstunden als effektive Alarmanlage eingesetzt werden. Wenn Sie nun den Spät-Krimi in ARD oder ZDF sehen wollen, so hat der Computer zwar die Möglichkeit, die Fenster und Türen Ihrer Wohnung zu Überwachen, eventuelle Störungen kann er aber nur durch einen zusätzlichen Verstärker akustisch melden. Durch die Verwendung eines Cassetten-Portsteckers, eines Widerstands, eines Lautsprechers und eines kleinen Programms erhalten Sie eine Alarmsirene, die auch den tiefsten Schlaf nachhaltig stört und im ausgeschalteten Zustand keinen Strom benötigt.



Das verwendete Prinzip besteht einfach aus schnellem Ein- und Ausschalten der Motorspannung für die Datasette. Da die Spannung an diesem Ausgang mit maximal 500 mA belastet werden kann, muß für einen Lautsprecher von 4 bis 8 Ohm Impedanz ein Widerstand in Reihe gelegt werden, der den Strom begrenzt. Dadurch wird zwar die Lautstärke geringer, aber noch bei 200 mA wird bei Resonanzfrequenz des Lautsprechers ganz schön Krach produziert. Als kleinster Vorwiderstand kommt bei einem 8 Ohm Lautsprecher ein 10 Ohm Widerstand in Frage. Statt des Lautsprechers kann auch eine Telephonhörerkapsel verwendet werden. Da der Widerstand dieser Kapseln bei etwa 200 Ohm liegt, kann dabei auf den Vorwiderstand verzichtet werden
Leider ist Basic für die Erzeugung von Frequenzen über 50 Hertz nicht verwendbar. Mit der Programmzeile

```
10 POKE 1 , 23 : POKE 1 , 55 : GOTO 10
```

entlocken wir dem Lautsprecher nur einen Brummton. Darum muß die Tonerzeugung in Maschinensprache erfolgen. Das benötigte Programm ist aber nur wenige Bytes lang und dürfte auch von 'Nur' Basic-Programmierern verstanden werden.

```
033C                    *=    $033C
033C           PORT     =     1
033C           EIN      =     %010111
033C           AUS      =     %110111
033C           DELAY    =     $F7
033C 78                 SEI
033D A0 00              LDY   #0
033F A6 F7     BEEP     LDX   DELAY
0341 A9 17              LDA   #EIN
0343 85 01              STA   PORT
0345 CA        LOOP1    DEX
```

```
0346 D0 FD              BNE   LOOP1
0348 A6 F7              LDX   DELAY
034A A9 37              LDA   #AUS
034C 85 01              STA   PORT
034E CA        LOOP2    DEX
034F D0 FD              BNE   LOOP2
0351 88                 DEY
0352 D0 EB              BNE   BEEP
0354 58                 CLI
0355 60                 RTS
```

Dieses kleine Programm liegt im Cassettenpuffer. Dieser Bereich kann ruhig benutzt werden, da die Datasette ja nicht betrieben werden kann, wenn der Lautsprecher angeschlossen ist.
Zur Funktion des Programms ist folgendes zu Sagen.
Zuerst werden Interrupts ausgeschaltet. Dies ist nicht unbedingt erforderlich, wenn aber darauf verzichtet wird, so ist der erzeugte Ton sehr unregelmäßig. Das Y-Register wird als Periodenzähler verwendet. Der Wert des X-Registers wird aus der Speicherzelle 247 geholt. In diese Speicherzelle kann ein Wert gePOKEd werden, der die Tonhöhe bestimmt. Je kleiner der Wert dieser Speicherzelle ist, desdo höher ist der erzeugte Ton.
Um die Handhabung des Programms zu erleichtern folgt jetzt ein Basic-Loader und ein kleines Beispielprogramm.

```
100 FOR I = 828 TO 853
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 120,160,  0,166,247,169, 23,133,  1,202,208,253
130 DATA 166,247,169, 55,133,  1,202,208,253,136,208,235
140 DATA  88, 96
150 IF S <> 3879 THEN PRINT "FEHLER IN DATAS !!" : END
160 PRINT "OK"
```

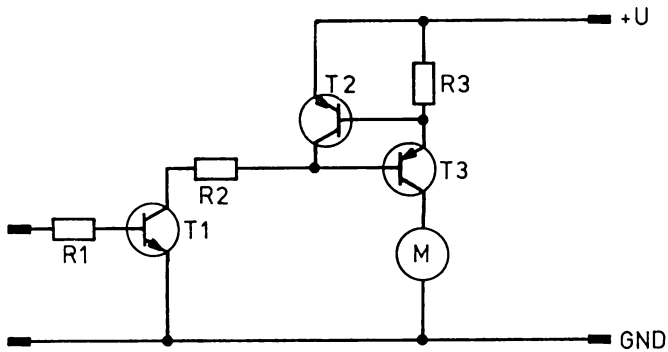
Nachdem dieses Programm gestartet wurde, kann mit dem folgenden Programm probiert werden, wie sich unterschiedliche Werte in der Speicherzelle 247 auf die Tonhöhe auswirken.

```
100 PRINT CHR$(147)
110 PRINT "BITTE WERT ZWISCHEN 1 UND 255 EINGEBEN"
120 INPUT A
130 IF A = 0 THEN END
140 POKE 247,A
150 SYS 828
160 GET A$ : IF A$ <> "" THEN GOTO 100
170 FOR I = 1 TO 200 : NEXTI
180 GOTO 150
```

Nach dem Eingeben eines Wertes für die Tonhöhe wird mit dem SYS das Maschinenspracheprogramm aufgerufen. Wenn der Motorausgang 256 mal ein und ausgeschaltet worden ist, wird mit GET in der Zeile 150 überprüft, ob eine beliebige Taste gedrückt ist. Wenn keine Taste gedrückt ist, wird eine kurze Zeit gewartet und danach wieder die Routine im Kassettenpuffer aufgerufen.
Haben Sie aber eine Taste gedrückt, so stoppt die Tonerzeugung, und Sie können einen neuen Frequenzwert eingeben. Wird hierfür 0 gewählt, so wird das Programm beendet.

4.8.1 Motorsteuerung Gleichstrommotoren

Die Ansteuerung von Gleichstrommotoren kann im einfachsten Fall mit einer Transistorstufe nach Kapitel 4.2 erfolgen. Besteht aber die Gefahr, daß der Motor schwer anläuft oder eventuell blockiert, dann ist diese einfache Schaltung nicht mehr ausreichend, da in diesem Fall der Motorstrom schnell Größenordnungen erreicht, der das entgültige Abschalten des Transistors bewirkt. Besser geeignet ist da eine Schaltung mit Strombegrenzung.



	15 V	30 V
R1	22 K	22 K
R2	4K7	10 K
R3	siehe Text	
T1	BC 237	
T2	BC 337	
T3	je nach max. Strom Kühlkörper !	

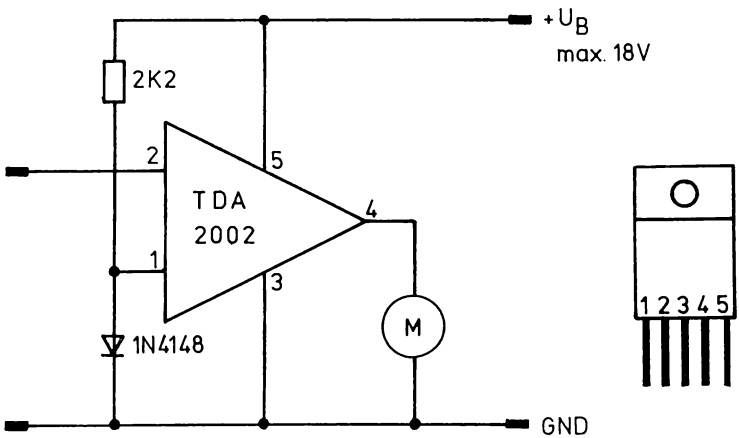
Sobald die Basis von T1 angesteuert wird, beginnt der Motor zu laufen. Am Widerstand R3 tritt ein zum Motorstrom proportionaler Spannungsabfall auf. Wenn jedoch durch

Überlast der Motor einen zu hohen Strom aufnimmt, so sinkt die Spannung an der Basis von T2. Wird die Spannung an der Basis um 0,7 V kleiner als die Betriebsspannung, dann beginnt T2 zu leiten und die Basis von T3 wird positiver. Dadurch sperrt T3 und der Motor bèkommt einen geringeren Strom. Der maximale Strom ist vom Wert des Widerstands R3 abhängig. Berechnet wird der Widerstand mit der Formel:

R3 = 0.7 / Imax.

Wird als maximaler Strom z.B. 300 mA zugelassen, so muß der Widerstand rechnerisch einen Wert von 2.3333333 Ohm haben, praktikabel ist ein Wert von 2.2 Ohm.

Für Motoren mit einer Stromaufnahme bis zu 3.5 Ampere und Spannungen bis 18 Volt lässt sich mit dem IC TDA 2002 eine sehr einfache und zuverlässige Motoransteuerung aufbauen.

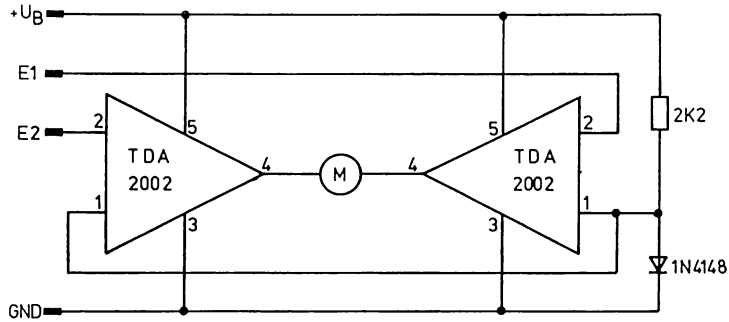


Obwohl ursprünglich für NF-Endstufen konstruiert eignet sich dies IC herforragend für unseren Zweck. Der Ausgang ist kurzschlußfest, kurzzeitige Spannungsspitzen bis 40 Volt werden klaglos überstanden und eine thermische Sicherung schützt das IC auch unter extremen Belastungen.
Außer dem IC werden nur noch ein Widerstand und eine Diode benötigt. Die Schaltung ist in der angegebenen Dimensionierung TTL-Kompatibel. Allerdings muß beachtet werden, daß die Schaltung invertiert. Bei einem High am Eingang des Verstärkers stoppt der Motor, bei einem Low dreht er sich.

Sollte der verwendete Motor ein eisenloser Glockenankermotor sein, so kann es bei einigen Exemplaren passieren, daß der Motor im ausgeschalteten Zustand ganz langsam weiterdreht. Das liegt an dem Umstand, daß der Ausgang des Verstärkers nicht exakt auf 0 Volt heruntergeregelt wird. Die minimale Ausgangsspannung beträgt etwa 0.15 Volt, genug um Motoren des genannten Typs im Bewegung zu halten. In diesem Fall kann in Reihe zum Motor die gestrichelt eingezeichnete Diode eingebaut werden. Der Wert der Diode richtet sich natürlich nach der Stromaufnahme des Motors. Bis 1 Ampere genügt eine 1N4001, bei größeren Strömen sollte eine 3 Ampere-Diode (1N5401 oder ähnlich) eingesetzt werden.

Mit den bisher gezeigten Schaltungen kann der Motor nur in einer Richtung gedreht werden. Für Positionieraufgaben ist aber häufig ein Betrieb in beide Drehrichtungen nötig. Für eine solche Anwendung ist eine Brückenschaltung erforderlich. Mit zwei ICs TDA 2002 ist so eine Brückenschaltung aber leicht realisiert. Das Schaltbild zeigt eine erprobte Anwendung, die den Motor wahlweise links oder rechts herum laufen lässt. Wie Sie sehen, sind für diese Schaltung zwei Port-Bits nötig. Um den Motor zu stoppen, werden beide Bits gelöscht. Um den Motor in Bewegung zu versetzen wird je nach

gewünschter Drehrichtung das eine oder das ander Bit gesetzt.



4.8.2 Motorsteuerung Schrittschaltmotoren

Schrittmotoren unterscheiden sich grundsätzlich von den bekannten Gleichstrommotoren. Mit Schrittmotoren lassen sich elektrische Impulse direkt in eine definierte Winkeländerung der Achse umsetzen. Bei jeder Ansteuerung mit den entsprechenden Signalen wird die Achse des Motors einen bestimmten von der Bauart des Motors abhängigen Betrag weiter gedreht. Für diese Ansteuerung besitzt der Motor vier Spulen.
Der große Vorteil des Schrittmotors ist, daß zu jeder Zeit die Position der Achse bekannt ist. Aus diesem Grund werden diese Motoren in der Microelektronik so gerne eingesetzt.
Bestes Beispiel für die Anwendung dieser Bauteile sind Floppy und Drucker.

In der Floppy wird die Kopfpositionierung mit einem solchen Motor vorgenommen. Das erklärt auch das klappernde Geräusch bei einigen Operationen des Laufwerks. In diesem Fall wird der Motor für den Kopftransport einfach einige Zeit in Richtung auf die Spur 0 zu bewegt. Danach ist sichergestellt, daß der Kopf die Ausgangsposition erreicht hat. Von dieser Startposition braucht jetzt nur die Anzahl der Schritte gezählt werden. Damit ist immer die genaue Position des Kopfes bekannt.

Bei Druckern werden Schrittmotoren für den Papiertransport und die Druckkopfbewegung eingesetzt. Auch in diesem Fall ist das Problem der exakten Positionierung einfach durch das Zählen der Schritte gelöst.

Wie geschieht nun die Ansteuerung im Einzelnen?

Dazu kann man 3 gebräuchliche Betriebsarten unterscheiden.

In der ersten möglichen Betriebsart werden die Spulen nacheinander mit einem Stromimpuls versorgt. Mit jedem Impuls dreht der Motor einen Schritt weiter.

Die zweite Betriebsart steuert immer zwei aufeinander folgende Spulen gleichzeitig, also Spule 1 und 2, dann 2 und 3, danach 3 und 4 und dann 4 und 1. Danach beginnt die Ansteuerung von neuem bei 1 und 2. Dieses Verfahren benötigt doppelt so viel Leistung, der Motor dreht aber ruhiger.

Das dritte übliche Verfahren besteht aus einer Kombination der beiden vorangegangenen Methoden. Hierbei wird zuerst nur Spule 1 angesteuert, dann Spule 1 und 2 gemeinsam, danach nur Spule 2, dann Spule 2 und 3 gemeinsam und so fort.

Wenn man sich die geschilderten Verfahren anschaut, so ist das zweite Verfahren sehr einfach programmtechnisch zu realisieren. Dazu wird der Motor in diesem Fall mit vier Portbits des User-Ports verbunden. Um den Motor zu bewegen, muß nur das Bitmuster 00110011 im Akku rotiert werden und an den User-Port ausgegeben werden.

Das bisher geschilderte Verfahren ist zwar möglich, aber

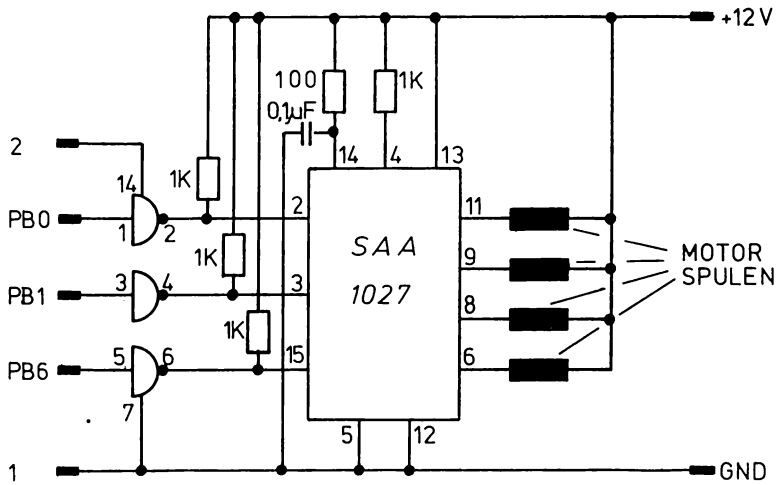
nicht besonders praktikabel. Der Softwareaufwand ist doch sehr groß. Von Basic aus lässt sich der Motor nicht ansteuern, es ist eine Programmierung in Maschinensprache notwendig.

Durch den Einsatz spezieller Ansteuer-ICs kann so ein Schrittmotor aber auch in Basic programmiert werden. Ein geeignetes IC ist das SAA 1027. Dieses IC erlaubt den direkten Betrieb von Schrittmotoren bis 350 mA Spulenstrom.

Sehen wir uns zunächst eimal die Schaltung einer Ansteuerstufe mit diesem IC an.Die drei mit dem User-Port verbundenen Pins mit den Bezeichnungen R, S und T haben die folgende Bedeutung:

R ist der Richtungseingang. Ein High an diesem Eingang lässt den Motor in die eine Richtung drehen, ein Low bewirkt Drehen in die entgegengesetzte Richtung.

S ist der Start-Stoppeingang. Bei einem High an diesem Eingang startet den Motor.



T ist der Takt- oder Triggereingang. Jedes Taktsignal an diesem Eingang bewirkt bei High-Pegel am S-Eingang ein Drehen der Achse. Die Drehrichtung wird dabei durch das am R-Eingang anliegende Signal festgelegt.
Der Eingang T ist verbunden mit PB6. An diesem Anschluß können von Timer A erzeugte Signale abgegriffen werden. Der Timer wird in der Art programmiert, daß an PB6 fortlaufend Signale auftreten. PB0 und PB1 sind verbunden mit den Eingängen S und R. Über PB0 kann nun bestimmt werden, ob der Motor dreht, PB1 bestimmt die Drehrichtung.
Allerdings ist beim Betrieb dieser Schaltung eine Besonderheit zu beachten, die in der Konstruktion des SAA 1027 begründet ist. Wird der Setzeingang S auf Masse gelegt, so stoppt, wie bereits erwähnt, der Stepper-Motor. Das bedeutet allerdings nicht, daß der Motor stromlos wird. Die Ausgänge des ICs nehmen vielmehr einen definierten Zustand ein. So werden die beiden Ausgänge Pin 6 und 9 Low. Damit fließt durch die entsprechenden Motorspulen der maximale nur durch den Spulenwiderstand begrenzte Strom. Bei Motoren mit sehr geringem Spulenwiderstand kann nach einiger Zeit durch zu starken Strom und damit verbundener Erwärmung eine Beschädigung des Motors auftreten. Es gibt glücklicherweise eine einfache Methode, alle Spulen stromlos zu schalten. Dazu wird nur ein weiterer Inverter mit Open-Collector-Ausgang benötigt. Den Ausgang legen Sie an Pin 4 des SAA 1027. Bei einem High am Eingang dieses Inverters wird die an Pin 4 liegende Spannung kurzgeschlossen und an allen Ausgängen liegt ein High. Dadurch fießt kein Spulenstrom mehr. Dieser zusätzliche Inverter ist im Schaltplan nicht eingezeichnet.
Ein vollständiges Basicprogramm zur Ansteuerung des Motors könnte etwa so aussehen:

```
100 S = 20
110 C2 = 56576
120 PB = C2+1
130 BR = C2+3
140 AL = C2+4
150 AH = C2+5
160 CA = C2+14
170 POKE 650,255
180 POKE BR,255
190 POKE AH,20
200 POKE AL,0
210 POKE CA,23
220 PRINT CHR$(147):PRINT:PRINT:PRINT
230 PRINT "      STEPPERMOTOR
240 PRINT :PRINT"   F1/F2 SCHALTET MOTOR EIN/AUS
250 PRINT :PRINT"   F3/F4 MOTOR LINKS/RECHTS
260 PRINT :PRINT"   F5/F6 MOTOR SCHNELLER/LANGSAMER
270 PRINT :PRINT"   F7    PROGRAMMENDE
280 GET A$ : IF A$="" THEN 280
281 IF ASC(A$)<133 OR ASC(A$)>139 THEN GOTO 280
282 A = ASC(A$) - 132
283 ON A GOSUB 290, 310, 330, 350, 300, 320, 340
284 GOTO 280
290 POKE PB,PEEK (PB) AND 254  :RETURN
300 POKE PB,PEEK (PB) OR 1     :RETURN
310 POKE PB,PEEK (PB) AND253   :RETURN
320 POKE PB,PEEK (PB) OR2      :RETURN
330 S=S-1 :GOSUB370 :POKE AH,S :RETURN
340 S=S+1 :GOSUB370 :POKE AH,S :RETURN
350 POKE PB,1 :END
370 IF S<4   THEN S=4
380 IF S>255 THEN S=255
390 RETURN
```

Bei diesem Programm wird allerdings von der wichtigsten Eigenschaft des Schrittmotors kein Gebrauch gemacht. Die einzelnen Schritte des Motors werden nicht gezählt. Damit kann wie bei einem normalen Gleichstrommotor keine Aussage über die Position gemacht werden.
Wie von dieser Möglichkeit Gebrauch gemacht werden kann, zeigt das nächste Programm. Hier werden zwar die Stepimpulse in Basic erzeugt, das verwendete Prinzip läßt sich aber leicht in Maschinensprache umsetzen.

```
110 C2 = 56576
120 PB = C2+1
130 BR = C2+3
140 CA = C2+14
150 POKE 650,255
160 POKE BR,255
170 POKE CA,0
180 FOR I=1TO6 :LS$=LS$+CHR$(157) :NEXT
190 PRINT CHR$(147)
200 PRINT :PRINT :PRINT
210 PRINT "      STEPPERMOTOR"
220 PRINT :PRINT " LAUFRICHTUNG   (L/R)   ";
230 INPUT RI$ :GOSUB 280
240 PRINT :PRINT " ANZAHL DER SCHRITTE    ";
250 INPUT SC :GOSUB 310
260 PRINT :PRINT " INSGESAMT ZURUECKGELEGT       ";LS$;AZ
270 PRINT CHR$(19) :GOTO 200
280 IF RI$="L" POKE PB,0 :FL =-1
290 IF RI$="R" POKE PB,2 :FL = 1
300 RETURN
310 FOR I=1 TO SC
320 POKE PB,PEEK (PB) OR 64:POKE PB,PEEK (PB) AND 191
330 NEXT :AZ =AZ + (FL*SC) : RETURN
```

5.0 Anschluß weiterer Peripherie-ICs

Die am User-Port zur Verfügung stehenden Ein-Ausgabeleitungen reichen für viele Steuerungsaufgaben nicht aus. Denken wir uns nur einmal eine etwas größere elektrische Eisenbahn. Um die vielen Weichen, Schranken und Signale zu steuern, ist eine wesentlich größere Anzahl von Steuerleitungen nötig. Aber auch für Steuerungsaufgaben im Haushalt oder im Labor eines Hobbyelektronikers könnte der C64 ruhig mehr Leitungen zur Verfügung stellen.
Diesem Mangel läßt sich aber recht einfach abhelfen.
Zusätzliche Peripherie-ICs wie die CIA 6526, aber auch 6522 oder 8255 lassen sich mit einfachen Mitteln anschließen. Der Anschluß erfolgt über den Expansion-Port. Leider ist dieser Anschluß als Leiterplattenverbinder ausgeführt und eine entsprechende Leiterplatte ist, anders als User-Portstecker oder Stecker für Controller-Ports, nicht so einfach zu bekommen. Nach einigem Suchen ist uns aber eine einfache und preiswerte Lösung eingefallen.
Die bekannten und für Versuchsaufbauten gerne eingesetzten Lochrasterplatten sind auch als beidseitig mit Kupferstreifen kaschierte Ausführung erhältlich. Der Abstand der Kupferstreifen zueinander beträgt exakt 2.54 mm, genau dem vom Leiterlattenverbinder im C64 vorgeschriebenen Maß. Mit einer Laubsäge lässt sich nun die Leiterplatte auf das gewünschte Maß anpassen.
Eine so bearbeitete Leiterplatte kann direkt zum Aufbau der gewünschten Schaltung verwendet werden. Preiswerter ist aber die Anfertigung eines Adapterstücks aus einer solchen Leiterplatte. Dazu löten Sie eine mindestens 40-polige Steckverbindung auf die zuvor auf Maß gebrachte Leiterplatte. Jetzt haben Sie beim Aufbau der Schaltungen freie Hand. Die Verbindung zur Adapterkarte kann dann über ein mehrpoliges Kabel zur Steckverbindung erfolgen.

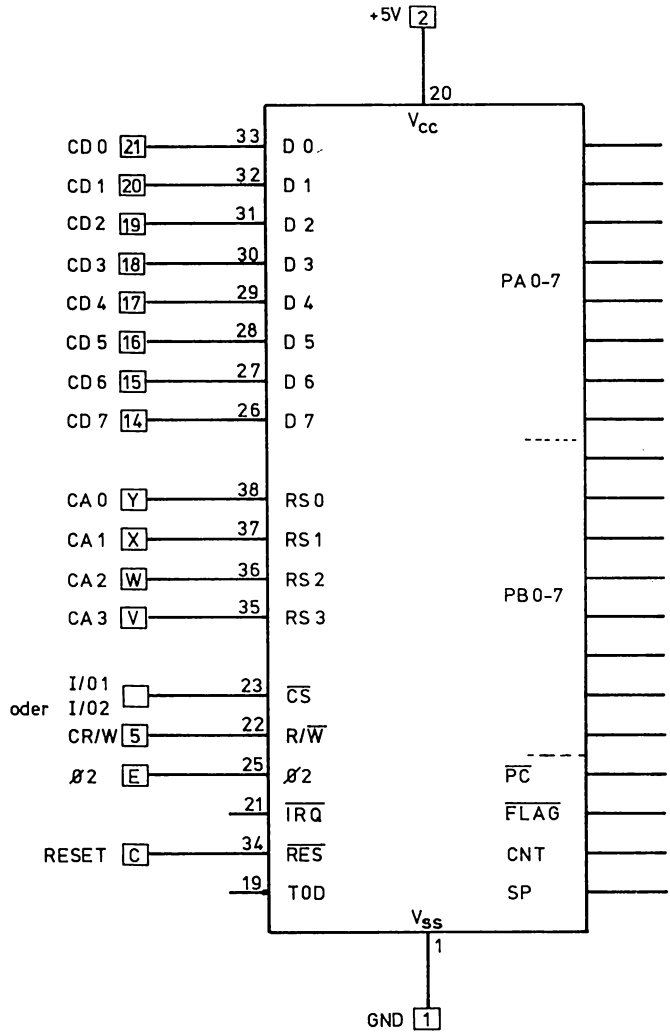
5.1 Anschluß weiterer CIAs 6526

Der Anschluß weiterer CIAs an den C64 bereitet keine Schwierigkeiten. Am Expansion-Port stehen uns ja alle benötigten Signale zur Verfügung. Das Signal -CS am Pin 23 der CIA wird entweder mit dem Signal I/O1 oder dem Signal I/O2 verbunden. Ob Sie I/O1 oder I/O2 benutzen, ist Ihnen freigestellt. Sie müssen nur die unterschiedlichen Basisadressen beachten. Haben Sie I/O1 benutzt, so liegt die Basisadresse auf $DE00 = 56832, bei I/O2 liegt sie auf $DF00 = 57088.

Ein auf diese Weise angeschlossener 6526 ist genau so über PEEK und POKE zu handhaben, wie die bereits eingebauten CIAs.

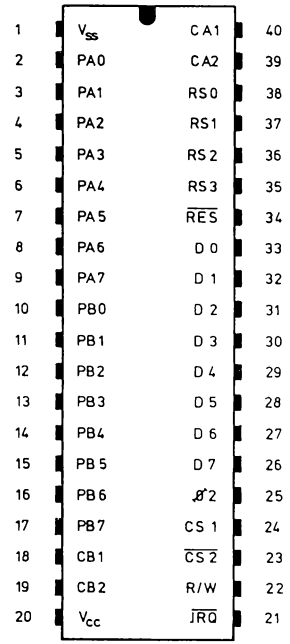
Sollen noch mehr dieser ICs angeschlossen werden, dann muß das -CS-Signal über weitere Adressdecoder gewonnen werden. Mit einem IC 74LS138 können dann maximal 8, mit zwei 74LS138 sogar 16 CIAs angeschlossen werden, beim derzeitigen Preis von etwa 40 DM kein ganz preiswertes Vergnügen. Die entsprechende Schaltung finden Sie im nächsten Kapitel beim Anschluß der VIA 6522.

Wenn Sie wirklich mehr Ein/Ausgabeleitungen für Steuerungen benötigen, sollten Sie sich allerdings nach preiswerteren Möglichkeiten umsehen. Eine gute Alternative ist dann die Verwendung der schon genannten 6522 oder des 8255. Den Anschluß dieser ICs werden wir nun betrachten.


+5V 2
20
Vcc
CD 0 21
CD 1 20
CD 2 19
CD 3 18
CD 4 17
CD 5 16
CD 6 15
CD 7 14
33 D 0
32 D 1
31 D 2
30 D 3
29 D 4
28 D 5
27 D 6
26 D 7
PA 0-7
CA 0 Y
CA 1 X
CA 2 W
CA 3 V
38 RS 0
37 RS 1
36 RS 2
35 RS 3
PB 0-7
I/01
oder I/02
23 CS
CR/W 5
22 R/W
ø2 E
25 ø2
21 IRQ
RESET C
34 RES
19 TOD
PC
FLAG
CNT
SP
Vss
1
GND 1

5.2 Die VIA 6522

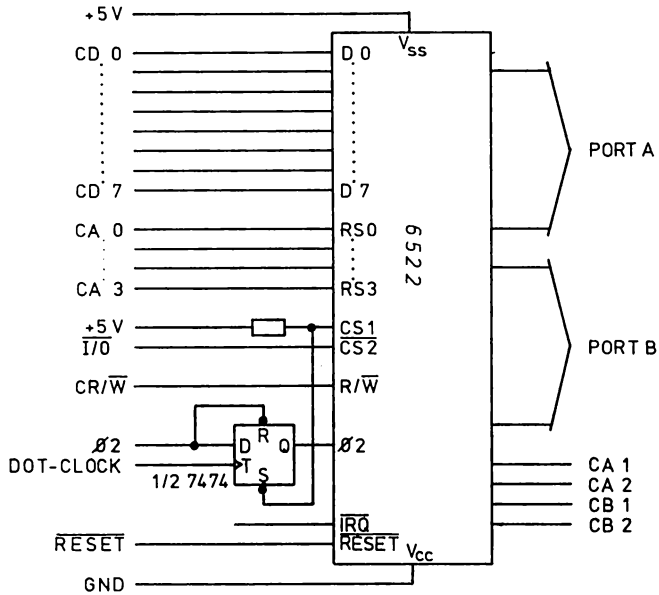
Die VIA 6522 (Versatile Interface Adapter) kann als der ältere Bruder des 6526 bezeichnet werden. Dies IC enthält 16 Register, über die zwei 8-Bit-Datenports, zwei 16-Bit-Timer, ein Schieberegister und verschiedene Steuerleitungen gehandhabt werden.
Wie man sieht, sind bis auf die Echtzeituhr TOD alle Möglichkeiten des 6526 eingebaut.
Die Handhabung des 6522 ist allerdings etwas komplizierter und nicht so logisch wie beim 6526. Allein der Preisunterschied und die bessere Verfügbarkeit der VIA wiegen diese Nachteile aber bei weitem auf.



Betrachten wir uns zuerst die Anschlußbelegung des ICs.
Der Port A kann je nach Programmierung als Ausgabe- oder als Eingabeport arbeiten. Dabei kann die Datenrichtung eines jeden Portbits einzeln programmiert werden. Bei der Verwendung als Eingabeleitung können die Eingabewerte im VIA gespeichert werden.
Dafür findet der Anschluß CA1 Verwendung. Mit jedem Low-High-Wechsel an CA1 werden die an den als Eingängen verwendeten Portbits anliegende Information im VIA gespeichert. Der CA1-Anschluß entspricht in dieser Funktion etwa dem -FLAG-Eingang des 6526. Durch Programmierung sind die beiden CA-Anschlüsse

aber auch als Interrupteingänge verwendbar.
Port B ist ebenfalls in seiner Datenrichtung an jedem Portbit frei programmierbar. Auch hier können die an den als Eingang programmierten Leitungen liegenden Informationen mit einem Hand-Shake-Signal im VIA gespeichert werden. Zusätzlich ist das Bit PB7 in der Art zu programmieren, daß dieser Anschluß den Ausgang eines der beiden integrierten Timer darstellt. PB6 dagegen ist als Eingang für den anderen Timer programmierbar.
Bei entsprechender Programmierung arbeitet CB1 als Hand-Shake-Eingang. Aber die Anschlüsse CB1 und CB2 können auch in anderen Betriebsarten betrieben werden. Ebenso wie die CA-Leitungen können sie zum einen als Interrupteingänge verwendet werden. Zum zweiten können diese Leitungen aber den Ein- und Ausgang des Schieberegisters darstellen. Damit eignen sich diese Leitungen auch zur Realisation serieller Datenübertragung.
Der Ausgang -IRQ wird normalerweise zur Erzeugung eines Interrupts verwendet. Dieser Anschluß wird immer dann Low, wenn bestimmte programmierte Ereignisse eintreten. Dies kann zum Beispiel in Verbindung mit den Timern, dem Schieberegister oder auch den Portcontrollanschlüssen geschehen. In unserem Beispiel ist der -IRQ-Pin unbeschaltet, er kann bei Bedarf wahlweise auf den NMI- oder IRQ-Anschluß des Expansion-Ports gelegt werden.
Über den R/-W-Anschluß signalisiert der Prozessor, ob er Daten in eines der 16 Register des 6522 schreiben oder aus dem VIC lesen will. Mit diesem R/-W gibt es eine Besonderheit, auf die wir sofort zu sprechen kommen.
Der Datentransfer vom Prozessor zu den VIAs geschieht über die Datenleitungen D0 bis D7.
Die Adressierung der 16 internen Register einer VIA ist etwas aufwendig. Für diesen Zweck verfügt ein 6522 über sechs Pins. Die eigentliche Registerauswahl wird mit den

vier RS- oder Register Select-Pins vorgenommen. Diese Anschlüsse sind normalerweise mit den unteren vier Adressbits des Prozessors verbunden. Damit ergibt sich eine durchgehende Adressierung der Register. Zusätzlich hat jeder 6522 zwei CS- oder Chip Select-Eingänge. Der CS1 ist ein High-aktives Signal, -CS2 ist Low-aktiv. Um also eine VIA zu selektieren, muss das Signal CS1 High, -CS2 dagegen gleichzeitig Low sein.



Obwohl auch dieses IC der Familie der 65xx-Bausteine angehört, ergeben sich doch Schwierigkeiten beim Anschluß an den C64. Die Ursache ist im Timing des C64 zu suchen. Für eine ordnungsgemäße Funktion muß das Taktsignal 02 geringfügig gegenüber dem Read/-Write Impuls verzögert werden.

Eine erprobte Schaltung zum Anschluß eines 6522 zeigt die nachstehende Abbildung. Die Verzögerung des R/-W erfolgt über ein Flip-Flop 74LS74 und dem Signal Dot-Clock des Rechners. Das Signal Dot-Clock des C64 hat die 8-fache Frequenz des 02. Sobald das Signal 02 des C64 High wird, wird mit der nächsten ansteigenden Flanke vom Dot-Clock der Ausgang des Flip-Flops High. Dadurch erhalten wir eine geringe aber ausreichende Verzögerung von 02. Leider würde nach dieser Methode auch die abfallende Flanke des auf diese Weise behandelten 02 verzögert. Darum wird der 02 der Rechners auch auf den CLEAR-Eingang des Flip-Flops geführt. Sobald 02 LOW wird, wird damit auch das Flip-Flop zurückgesetzt.

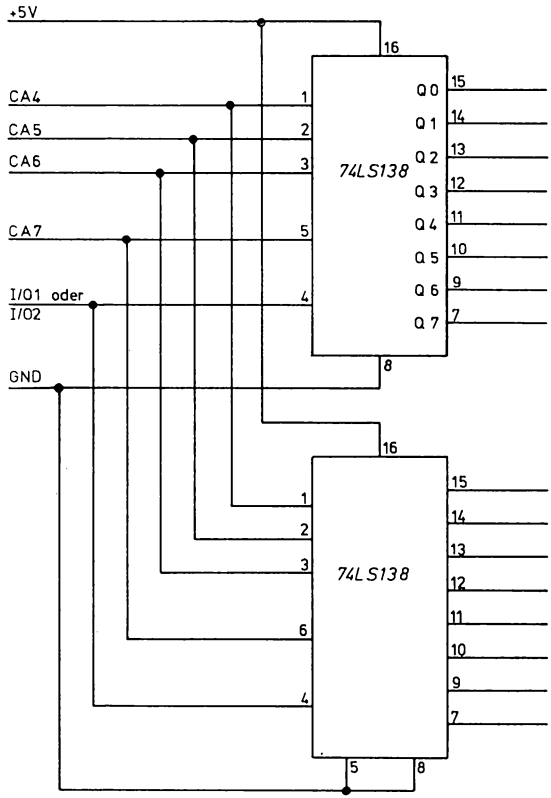
Wie im Schaltplan zu sehen, ist der Eingang -CS mit dem Signal -I/O 1 verbunden. Damit belegt die VIA den Adressbereich von $DE00 bis $DEFF. Jedes der 16 internen Register erscheinen somit jeweils 16 mal in diesem Adressbereich. Wenn die durch die VIA zusätzlich verfügbaren Leitungen ausreichen, ist dagegen nichts einzuwenden. Reicht aber eine VIA nicht aus, dann kann die nachstehende Schaltung verwendet werden, um bis zu 16 VIAs anzuschließen. Damit stehen dann bis zu 256 I/O-Leitungen gleichzeitig zur Verfügung.

Die Schaltung ist aus zwei Dekodern 74LS138 aufgebaut. An den Ausgängen erhalten Sie Low-aktive Signale für jeweils einen Adressbereich von 16 Bytes. So ist beispielsweise der Ausgang Q0 des oberen Adressdekoders im Bereich $DE00 bis DE0F Low, der Ausgang Q3 des unteren Dekoders Low im Adressbereich von $DEB0 bis $DEBF.

Da der C64 auch für den Adressbereich $DF00 bis $DFFF ein I/O-Signal, das I/O2, bereitstellt, läßt sich die Anzahl der Portleitungen noch einmal verdoppeln.

In den meisten Fällen wird aber eine solch große Anzahl von Periperie-ICs nicht nötig sein, so daß auf den unteren

Adressdekoder auch verzichtet werden kann. Dann reduziert sich die Anzahl anschließbarer 6526 und 6522 auf acht, wobei natürlich bei Bedarf ein gemischter Betrieb möglich ist.



Doch betrachten wir nun die Registerbelegung des 6522.

REG 0 Datenregister B (ORB)
Dieses Register gibt den Zustand des Datenport B wieder. Es kann sowohl geladen als auch gelesen werden.
Achtung: Ein Zugriff auf dieses Register setzt die Bits 3&4 des IFR (REG 13) zurück und kann, abhängig vom Zustand der Bits 5-7 im PCR (REG 12), die Leitung CA2 beeinflussen!

REG 1 Datenregister A (ORA)
Hiermit wird der Datenport A gehandhabt, Zugriffsart Lesen/Schreiben.
Achtung: Die Bits 0&1 des IFR (REG 13) werden zurückgesetzt. Außerdem kann, abhängig vom Zustand der Bits 1-3 im PCR (REG 12), die Leitung CA2 beeinflußt werden.

REG 2 Datenrichtung Port B (DDRB)
Mittels dieses Registers können die Leitungen des Datenport B individuell auf Eingabe oder Ausgabe programmiert werden. Ist ein Bit dieses Registers =0, so steht die korrespondierende Datenleitung auf Eingabe; ist es =1, steht die Leitung auf Ausgabe.

REG 3 Datenrichtung Port A (DDRA)
Bedeutung wie REG 2, allerdings für den Datenport A.

REG 4 Timer 1 LO (T1CL)
Beim Lesezugriff gibt dieses Register den Zustand des niederwertigen Teiles von Timer 1 wieder. Beim Laden wird zunächst nur ein Zwischenspeicher mit dem neuen Wert versorgt. Die Übernahme in den Zähler

erfolgt erst mit Schreiben des höherwertigen Bytes (REG 5).

REG 5 Timer 1 HI (T1CH)

Beim Lesen erhalten Sie den höherwertigen Teil von Timer 1. Ein Schreibzugriff lädt sowohl den Zwischenspeicher als auch den Zähler. Gleichzeitig wird das niederwertige Byte aus dem Zwischenspeicher in den Zähler übernommen.

Achtung: Bit 6 des IFR (REG 13) wird beim Schreiben zurückgesetzt!

Abhängig von Bit 6&7 des ACR (REG 11) kann die Leitung PB7 und das Bit 6 des IFR (REG 13) beeinflußt werden.

ACR6&7 =00:

Beim Nulldurchgang des Zählers wird IFR6 gesetzt. Der Timer zählt zwar nun erneut bis Null, jedoch kann IFR6 erst wieder nach dem Schreiben von REG 5 gesetzt werden.

ACR6&7 =01:

Wie vor, jedoch wird PB7 beim Laden von REG 5 =LO und beim nächsten Nulldurchgang =HI. PB7 steht unabhängig von DDRB auf Ausgang!

ACR6&7 =10:

Der Timer zählt zyklisch vom einmal geladenen Wert auf Null, jedoch wird IFR6 nur beim ersten Nulldurchgang nach Laden von T1CH gesetzt. IFR6 kann durch Lesen von T1CL, Schreiben von T1CH oder explizit, wie bei REG 13 beschrieben, rückgesetzt werden.

ACR6&7 =11:

Wie vor, jedoch kippt PB7 bei jedem Nulldurchgang in die jeweils andere Lage.

REG 6&7 Timer 1 LO&HI (T1LL&T1LH)

Vom Zugriff auf diese Register sind sowohl beim Schreiben als auch beim Lesen grundsätzlich nur die Zwischenspeicher der jeweiligen Zähler betroffen. Die Zähler selbst werden nicht verändert. Interessant ist dieser Effekt, wenn die Timer im FREE-RUNNING-MODE (REG 11 Bit 6 =1) betrieben werden. Hierbei wird bei jedem Nulldurchgang des Zählers dieser automatisch mit dem Inhalt des Zwischenspeichers geladen.
Achtung: Auch beim Schreibzugriff auf REG 6 wird Bit 6 des IFR (REG 13) zurückgesetzt!

REG 8&9 Timer 2 LO&HI (T2CL&T2CH)

Für diese Register trifft bezüglich Lesen und Laden sinngemäß die Beschreibung der REG 4&5 zu. Allerdings gibt es hier nur zwei Betriebsarten, abhängig von Bit 5 im ACR (REG 11)
ACR5 =0:
Sinngemäß wie bei Timer 1, Betriebsart 000 beschrieben; allerdings ist hier Bit 5 des IFR (REG 13) vom Nulldurchgang betroffen.
ACR5 =1:
Ein in den Timer 2 geladener Wert wird durch von außen an PB6 angelegte Impulse auf Null gezählt, wobei beim Nulldurchgang IFR5 gesetzt wird. Der Timer zählt nun erneut auf Null, jedoch kann IFR5 erst wieder beim dem Laden von REG 9 folgenden Nulldurchgang gesetzt werden.
Achtung: Ein Schreibzugriff auf REG 9 setzt Bit 5 des IFR (REG 13) zurück!

REG 10 Schieberegister (SR)

Das Schieberegister kann ohne weiteres geschrieben und gelesen werden. Die hierbei ausgelösten Vorgänge

sind abhängig vom Zustand der Bits 2-4 des ACR (REG 11).

ACR2-4 =000:

Das Schieberegister kann ohne weitere Auswirkungen geschrieben und gelesen werden.

ACR2-4 =100:

Die an CB2 anliegenden Daten werden mit einer durch T2CL (REG 8) bestimmten Geschwindigkeit in SR hineingeschoben. Der Schiebetakt kann durch die Formel CLK/2/n berechnet werden, wobei CLK die Systemtaktfrequenz und n der nach REG 8 geladene Wert ist. Dieser Schiebetakt steht auch an CB1 zur evtl. Synchronisation der Datenquelle zur Verfügung. Nach acht Impulsen wird der Schiebevorgang unterbrochen und erst fortgesetzt, wenn SR gelesen wurde. Außerdem wird Bit 2 des IFR (REG 13) gesetzt.

ACR2-4 =010:

Der Schiebetakt wird hier durch den Systemtakt erzeugt und beträgt CLK/2. Sonst wie vor.

ACR2-4 =110:

Der Schiebetakt muß in dieser Betriebsart von außen an CB1 zugeführt werden. Nach acht Takten wird auch hier IFR2 gesetzt, jedoch der Schiebevorgang nicht unterbrochen.

ACR2-4 =001:

Der Inhalt des Schieberegisters wird mit einer durch den Systemtakt und den Inhalt von REG 8 bestimmten Geschwindigkeit nach CB2 hinausgeschoben. Dieser Vorgang läuft zyklisch, d.h. die aus SR hinausgeschobenen Bits werden wieder hineingeleitet. Der Schiebetakt ist wieder an CB1 verfügbar.

ACR2-4 =101:

Der Schiebevorgang wird nach acht Takten abgebrochen und erst nach Laden des SR fortgsetzt. Außerdem wird

IFR2 gesetzt. Sonst wie vor.
ACR2-4 =011:
Der Schiebetakt beträgt CLK/2. Sonst wie vor.
ACR2-4 =111:
Die Schiebeimpulse müssen von außen an CB1 angelegt werden. Nach acht Takten wird zwar IFR2 gesetzt, jedoch der Schiebevorgang nicht unterbrochen.
Achtung: Für alle Betriebsarten gilt, daß nach acht Impulsen, gleichgültig ob intern erzeugt oder von außen angelegt, das Bit 2 im IFR (REG 13) gesetzt wird. Gelöscht wird dieses Bit durch Schreiben oder Lesen des SR.

REG 11 Hilfssteuerregister (ACR)

Die Bits dieses Registers werden hier nur soweit beschrieben, wie sie nicht bereits bei den Timern und dem Schieberegister behandelt wurden.
Bit 0 =0: Die aus ORA gelesenen Daten entsprechen dem aktuellen Zustand der Leitungen PA0-7.
=1: Der beim Setzen von IFR1 durch CA1 an PA0-7 vorhandene Zustand wird bis zum Löschen von IFR1 (REG 13) im ORA festgehalten.
Bit 1: Sinngemäß wie oben, jedoch sind hier ORB, PB0-7, CB1 und IFR4 betroffen.
Bit 2-4: Steuerung des Schieberegisters SR (REG 10)
Bit 5: Steuerung von Timer 2 (REG 8&9)
Bit 6-7: Steuerung von Timer 1 (REG 4&5)

REG 12 Port-Steuerregister (PCR)

Bit 0 =0: Eine fallende Flanke an CA1 setzt IFR1
=1: Eine steigende Flanke setzt IFR1
Bit 1-3=000: Eine fallende Flanke an CA2 setzt IFR0. IFR0 wird beim Zugriff auf ORA oder explizit zurückgesetzt.

=100: Wie oben, jedoch kann IFR0 nur explizit rückgesetzt werden.

=010: Wie bei =000, jedoch steigende Flanke.

=110: Wie bei =100, jedoch steigende Flanke.

=001: Zugriff auf ORA setzt CA2=LO. Nach einer aktiven Flanke an CA1 wird CA2 wieder HI.

=101: Nach einem Zugriff auf ORA wird CA2 für die Dauer eines Systemtaktes LO.

=011: CA2 wird LO.

=111: CA2 wird HI.

Bit 4 =0: Eine fallende Flanke an CB2 setzt IFR4.

=1: Eine steigende Flanke setzt IFR4.

Bit 5-7=000: Eine fallende Flanke an CB2 setzt IFR3, welches mittels Zugriff auf ORB oder explizit rückgesetzt werden kann.

=100: Wie oben, jedoch kann IFR3 nur explizit rückgesetzt werden.

=010: Wie bei =000, jedoch steigende Flanke.

=110: Wie bei =100, jedoch steigende Flanke.

=001: Ein Schreibzugriff auf ORB setzt CB2=LO. Eine aktive Flanke an CB1 setzt CB2 wieder HI.

=101: Nach einem Schreibzugriff auf ORB wird CB2 für die Dauer eines Systemtaktes LO.

=011: CB2 wird LO.

=111: CB2 wird HI.

REG 13 Interrupt-Flag-Register (IFR)

Die einzelnen Bits dieses Registers signalisieren das Eintreffen von Ereignissen, deren Eintrittsbedingungen hier nur noch global beschrieben werden, da sie im einzelnen bereits an ihren Ursprüngen erläutert wurden.

Bit 0: wird mit einer aktiven Flanke an CA2 gesetzt und mit einem Zugriff auf ORA (REG 1) rückgesetzt.

Bit 1: wird mit einer aktiven Flanke an CA1 gesetzt und mit einem Zugriff auf ORA (REG 1) rückgesetzt.
Bit 2: wird nach acht Schiebeimpulsen gesetzt und mit einem Zugriff auf SR (REG 10) rückgesetzt.
Bit 3: wird mit einer aktiven Flanke an CB2 gesetzt und mit einem Zugriff auf ORB (REG 0) rückgesetzt.
Bit 4: wird mit einer aktiven Flanke an CB1 gesetzt und mit einem Zugriff auf CB2 (REG 0) rückgesetzt.
Bit 5: wird beim Nulldurchgang von Timer 2 gesetzt und mit einem Lesezugriff auf T2CL (REG 8) oder einem Schreibzugriff auf T2CH (REG 9) rückgesetzt.
Bit 6: wird beim Nulldurchgang von Timer 1 gesetzt und mit einem Lesezugriff auf T1CL (REG 4) oder einem Schreibzugriff auf T1CH (REG 5) rückgesetzt.
Bit 7: wird gesetzt, wenn mindestens ein gesetztes Bit dieses Registers mit dem korrespondierenden gesetzten Bit im IER (REG 14) zusammentrifft. Dieses Bit gibt den Zustand der Leitung IRQ wieder.
Achtung: Die Bits 0-6 dieses Registers können auch explizit dadurch rückgesetzt werden, indem man das Register mit einem Byte lädt, dessen gesetzte Bits (=1) mit dem zu löschenden Bit dieses Registers korrespondieren.

REG 14 Interrupt-Enable-Register (IER)

Die Bits dieses Registers korrespondieren mit den Bits des IFR. Beeinflussung der Leitung IRQ wie unter IFR 7 beschrieben.
Die Bits dieses Register lassen sich setzen, indem man das Register mit einem Byte lädt, dessen gesetzte Bits mit dem zu setzenden Bit dieses Registers übereinstimmen, wobei das höchstwertige Bit (Bit 7) =1 sein muß. Die 0-Bits im Datenbyte lassen die korrespondierenden Bits dieses Registers unverändert.

Sollen einzelne Bits rückgesetzt werden, so ist der Vorgang wie oben beschrieben, jedoch muß hier das höchstwertige Bit des Datenbytes (Bit 7) =0 sein.

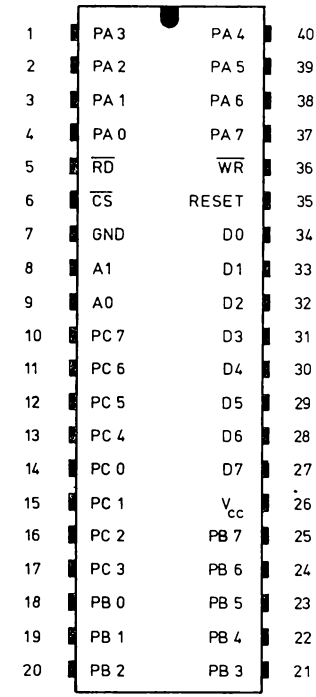
REG 15 Datenregister A (ORA)

Der Inhalt dieses Registers spiegelt die Datenleitungen PA 0-7 wieder, wie unter REG 1 beschrieben, jedoch hat der Zugriff in diesem Falle keinerlei Auswirkung auf die Steuerleitungen oder das IFR.

5.3 Der Parallelschnittstellenbaustein 8255

In einem System mit vielen I/O-Leitungen läßt sich vorteilhaft der 8255 einsetzen. Dieses IC gehört eigentlich in die Familie der Peripherie-ICs für 8080 und Z80.

Aber auch am C64 ist der Anschluß einfach und sogar unkritischer als dies beim 6522 der Fall ist. Dieses IC stellt dem Benutzer drei vollständige 8-Bit-Ports zur Vefügung.



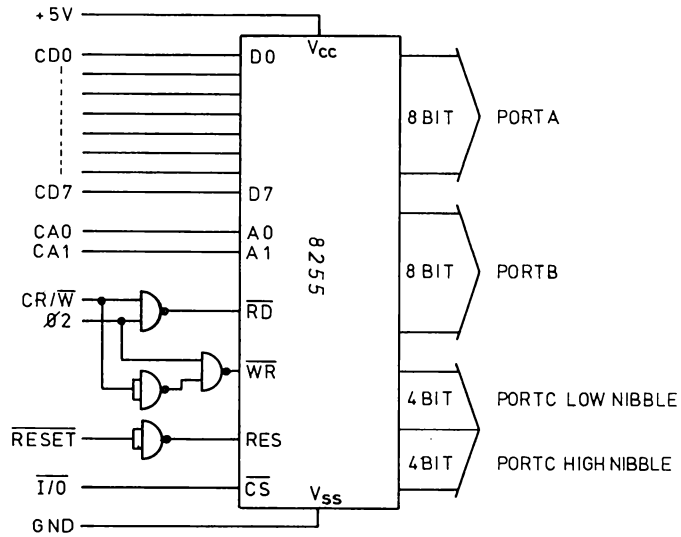
Da es als paralleles Schnittstellen-IC konzipiert ist, sind keine Timer oder Schieberegister integriert. Auch läßt sich die Datenrichtung nicht für jedes einzelne Portbit einstellen. Dafür benötigt das IC aber auch nur 4 Adressen. Diese Adressen besorgen die Registerauswahl an den Pins A0 und A1.

Der Anschluß -CS besorgt in der bekannten Art die Auswahl des ICs. Bei einem Low an diesem Eingang kann ein Datentransfer zwischen Prozessor und 8255 stattfinden.

Auf Grund seiner Zugehörigkeit zur Familie der 80er Prozessoren verfügt der 8255 über getrennte READ und WRITE-Anschlüsse. Durch die Verwendung von drei Gattern lassen sich die gewünschten Signale aber einfach aus dem R/-W-Signal des Prozessors generieren.

Der RESET-Eingang erfüllt beim 8255 die bekannten Aufgaben.

Durch ein High werden alle Register gelöscht und die Ports werden als Eingang programmiert. Zu beachten ist allerdings, daß die Polarität des vom C64 gelieferten RESET-Impulses invertiert werden muß.
Der Datentransfer von und zum Prozessor geschieht wie immer auf dem Datenbus über die Pins D0 - D7.
Die noch übrigen 24 Leitungen sind die Ports A, B und C. Ein realisierter Anschluß eines 8255 sieht folgendermaßen aus.



zusätzliche Gatter 1 IC 7400

Der Port A enthält einen Ausgabe-Zwischenspeicher/Puffer und einen Eingabe-Zwischenspeicher, Port B einen Ein/Ausgabe-Zwischenspeicher und einen Ausgabe-Zwischenspeicher. Port C enthält einen Ausgabezwischenspeicher und einen Eingabe-Puffer. Zusätzlich kann Port C als zwei 4-Bit-Ports programmiert werden, die dann als Hand-Shake-Leitungen für

Port A und B verwendet werden können. Das Low Nibble von Port C gehört dann zu Port B, das High Nibble zu Port A. Man spricht dann auch von Gruppe A und Gruppe B.

Grundsätzlich verfügt der 8255 über drei Betriebsarten.

ie Betriebsart 0 stellt eine einfache Ein-Ausgabe über die Portbits dar. Diese Betriebsart ist am besten zu vergleichen mit dem Betrieb der Ports der CIA 6526. In dieser Betriebsart können Schalter abgefragt werden oder Signale ausgegeben werden.

Die Betriebsart 1 ist ein sogenannter getasteter Betrieb. Diese Betriebsart ermöglicht komfortable Interrupt-steuerungen.

Betriebsart 2 ist ein bidirektionaler Ein/Ausgabe-Modus zwischen Datenbus und Port A.

Zusätzlich besteht die Möglichkeit, in dem als Ausgabe programmierten Port C gezielt Bits zu setzen oder zu löschen.

Wenn man sich diese ganzen Möglichkeiten einmal ansieht, so fragt man sich unwillkürlich, wie man eine solche Menge an Betriebsarten mit nur einem Steuerregister programmieren kann.

Um diese Steuerung zu realisieren, haben die Bits im Steuerregister bestimmte Bedeutungen. Maßgebend ist das Bit 7 im Steuerwort. Ist es gesetzt, dann haben die Bits 0 bis 6 die folgende Bedeutung.

Bit 0 Port C Low Nibble
1 = Eingang, 0 = Ausgang

Bit 1 Port B
1 = Eingang, 0 = Ausgang

Bit 2 Wahl der Betriebsart von Gruppe B
0 = Betriebsart 0, 1 = Betriebsart 1

Bit 3 Port C High Nibble
1 = Eingang, 0 = Ausgang

```
Bit 4    Port A
         1 = Eingang, 0 = Ausgang
Bit 5-6  Wahl der Betriebsart von Gruppe A
         Bit 6 5
             0 0 = Betriebsart 0
             0 1 = Betriebsart 1
             1 x = Betriebsart 2
```

Ist das Bit 7 dagegen gelöscht, sieht die Bedeutung so aus.

```
Bit 0    Bit setzen
         0 = Bit rücksetzen, 1 = Bit setzen
Bit 1-3  Bitauswahl
         Die Bitauswahl geschieht rein binär.
         Bit 3 2 1
             0 0 0 = Portbit0
             0 0 1 = Portbit1
                   :
                   :
             1 1 1 = Portbit7
Bit 4-6  ohne Bedeutung
```

Die Betriebsarten 1 und 2 werden nur selten genutzt. Auch wir werden uns auf die Betriebsart 0 beschränken. Zur Verdeutlichung sollen hier noch einige Beispiele für unterschiedliche Anwendungen folgen.

```
Port A = Ausgang, Port B = Eingang, Port C = Ausgang
Das Steuerwort lautet binär %10000010 = 130 dez.

Port A = Eingang, Port B = Ausgang,
Port C High Nibble = Ausgang
Port C Low  Nibble = Eingang
Das Steuerwort lautet binär %10011000 = 152 dez.
```

Ein wichtiger Punkt ist noch zu erwähnen. Wenn das Steuerwort neu geschrieben wird, dann werden alle an den als Ausgang gespeicherten Daten gelöscht. Auch kann das Steuerregister nur beschrieben, nicht aber gelesen werden.

5.4 Andere Peripherie-ICs

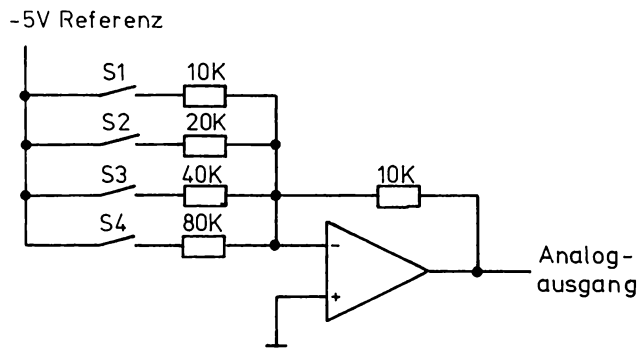
Die bisher beschriebenen Schnittstellenbausteine dürften für den C64-Besitzer die interessantesten ICs sein. Es gibt auf dem Markt aber noch eine große Anzahl anderer ICs mit zum Teil sehr interessanten Leistungsmerkmalen.
Die Palette der lieferbaren und an den C64 anschliessbaren Bauteile umfasst dabei weitere ICs zur Realisation von Parallelschnittstellen, genannt seien der 6520 und der Z80 PIO (Parallel IO). Aber auch Timer und Zähler wie 8253 oder Z80 CTC lassen sich mit den beschriebenen Interface-Methoden problemlos anschliessen.
Um eine bestimmte Anwendung zu lösen, ist ein genaues Studium der Datenblätter verschiedener Halbleiterhersteller wichtig. An Hand dieser Datenblätter kann dann der jeweils günstigste Baustein ausgesucht und eingesetzt werden.

6.0 Digital/Analog-Wandlung, was ist das?

Wie wir bis jetzt gesehen haben, kann ein Computer immer nur zwei Zustände signalisieren, entweder ist am Ausgang eine Spannung um 5 Volt zu messen, oder aber nicht. Dazwischen sind keine Werte möglich. Für viele Anwendungen sind aber Zwischenwerte nötig!
Nehmen wir als Beispiel nur einmal die Motorsteuerung aus dem Kapitel 4.8. Damit können wir den Elektromotor nur ein- oder ausschalten. Eine kontinuierliche Drehzahländerung ist mit dieser Schaltung nicht möglich. Um die Drehzahl regeln zu können, ist eine Schaltung nötig, die in Abhängigkeit von einem vom Computer gelieferten binären Wert einen entsprechenden Spannungwert liefert. Eine solche Schaltung ist als Digital/Analog-Wandler oder DAC (Digital-Analog Converter) bekannt.
Die gebräuchlichen DACs sind in Form von ICs erhältlich, die entweder eine Ausgangsspannung oder einen Ausgangsstrom erzeugen. Eine andere analoge Ausgangsgröße muß dann aus der Spannung oder dem Strom gewonnen werden. Doch betrachten wir zuerst einmal kurz den theoretischen Hintergrund, und sehen uns an, welche Arten der Digital/Analogwandlung es gibt.

6.1 Die Methoden der D/A-Wandlung

Es wird Ihnen wohl nicht bekannt sein, aber Ihr C64 enhält bereits mehrere DACs. Jedes Portbit des User-Ports ist im Grunde genommen ein DAC. Allerdings ist die Auflösung auf ein Bit begrenzt. Darum sind nur die Zustände 0 Volt und 5 Volt möglich. Das ist nicht gerade überragend, wir wollen ja gerade auch die verschiedenen Spannungen zwischen 0 und 5 Volt erzeugen. Damit ist schon klar, daß zu einer feineren Abstufung der Ausgangsspannung mehr als nur ein Bit nötig ist.
Mit vier Bits lassen sich 2↑4 = 16 verschiedenen Zustände erzeugen, also sollten sich auch 16 Ausgangsspannungen erzeugen lassen. Ein solcher (theoretischer) Vier-Bit-Wandler besteht aus einem Operationsverstärker, fünf Widerständen, vier Schaltern und einer Referenzspannung.



Die Funktion dieser Stufe ist schnell erklärt. Der Operationsverstärker (auch kurz OP genannt) arbeitet als Summierverstärker. Die Schalter lassen sich von einem Port aus schließen und öffnen. Sind alle Schalter geöffnet, so

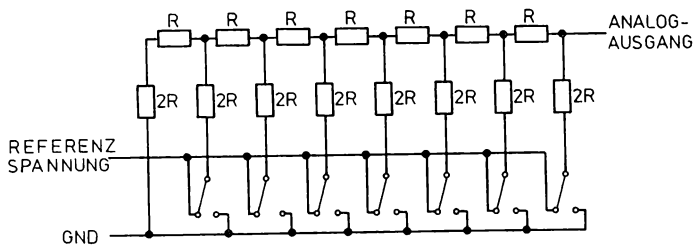
ist der Ausgang auf 0 Volt. Ist Schalter 1 geschlossen, so ergibt sich am Ausgang bedingt durch das Verhältnis von Eingangs- und Rückkopplungswiderstand eine Ausgangsspannung von 5 Volt. Ist dagegen Schalter 2 geschlossen, so erhalten wir eine Ausgangsspannung von nur 2.5 Volt. Bei geschlossenem Schalter 3 erhalten wir eine Spannung von 1.25 Volt, und wenn Schalter 4 gesclossen wird, dann ergibt sich wiederum die Hälfte der vorherigen Ausgangsspannung, genau 0.625 Volt.

Werden mehr als nur ein Schalter zur Zeit geschlossen, dann addieren sich die Eingangsspannungen. Werden alle Schalter geschlossen, dann ergibt sich somit eine maximale Ausgangsspannung von 9.375 Volt.

Dieses Prinzip kann nun auf mehr Bits erweitert werden. In der Praxis ist diese Art der D/A-Wandlung aber bei höherer Auflösung nicht sehr gut geeignet, da die Widerstände immer hochohmiger werden müssen. Dabei stösst man aber schnell an die im Herstellungsprozess gegebenen Grenzen.

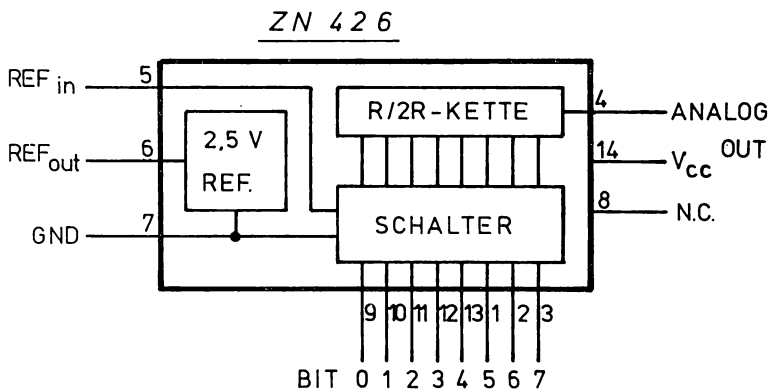
Darum hat man das verbreitete Prinzip der sogenannten R/2R-Widerstandsleiter ersonnen. Hier werden nur zwei Widerstandswerte benötigt, und das unabhängig von der Auflösung.

Die Abbildung zeigt einen Ausschnitt aus dem Innenleben eines solchen DACs.



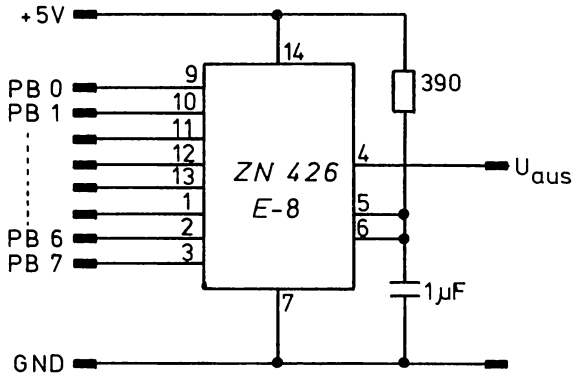
6.2 Der ZN 426 - ein universeller Wandler

Der ZN 426 ist ein praktischer Baustein, der sich sehr einfach an den User-Port anschließen läßt. Er enthält alle benötigten Stufen wie R/2R-Widerstandsleiter, Stromschalter und Referenzspannung. Auch die technischen Daten können sich sehen lassen. So beträgt die Referenzspannung 2.5 Volt und ist sehr temparaturstabil (40 ppm). Auch die Geschwindigkeit der Wandlung ist als sehr gut zu bezeichen. Die Ausgangsspannung liegt nach nur 1 Microsekunde an.
Die Auflösung von 8 Bit eignet sich natürlich hervorragend zum Anschluß an 8-Bit-Rechner. Als zusätzliche Beschaltung sind nur ein Widerstand und ein Kondensator für die Referenzspannung nötig.



Die maximale Ausgangsspannung liegt in der Höhe der Referenzspannung, also bei etwa 2.5 Volt. Da mit 8 Bit 256 Schritte realisierbar sind, beträgt der kleinste Spannungssprung ziemlich genau 10 Millivolt. Sind alle Bits auf Low, so ergibt sich eine Ausgangsspannung von weniger als 5 Millivolt, typisch sind 3 Millivolt.Daraus resultiert

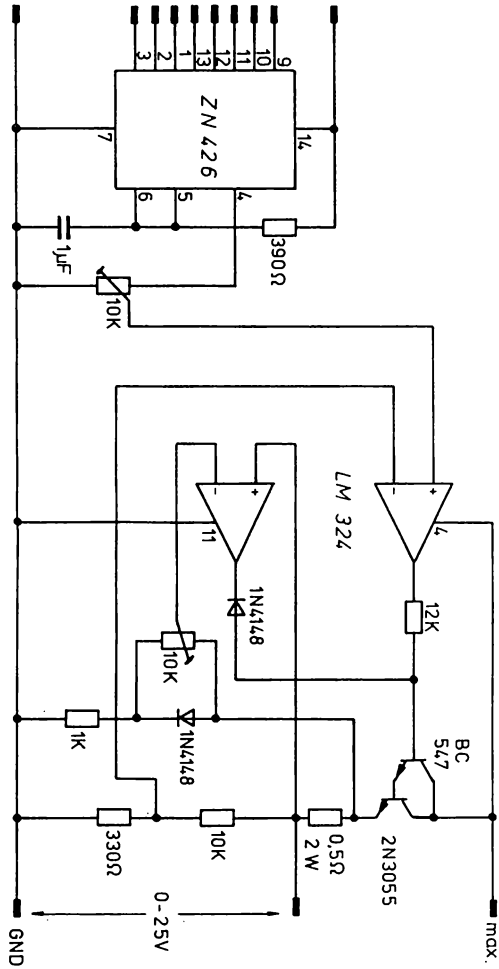
ein Fehler von 1/2 LSB.
Da der Ausgang des ZN 426 recht hochohmig ist, er beträgt etwa 10 Kiloohm, sollten Sie bei Bedarf eine Ausgangsstufe mit einem Operationsverstärker vorsehen.



Doch nach so viel Theorie wollen wir uns wieder der Praxis zuwenden. Unsere erste Anwendung des DAC ist eine Verwendung als programmierbare Stromversorgung.

6.2.1 Programmierbare Stromversorgung

Die hier vorgestellte Schaltung liefert eine Ausgangsspannung von fast 0 Volt an bis zu einer maximalen Spannung von 25.5 Volt. Damit entspricht jede Stufe des Wandlers einer Ausgangsspannung von 100 mV. Zusätzlich enthält die Ausgangsstufe eine einstellbare Strombegrenzung. In unserem Entwurf wird der Ausgangsstrom durch einen Trimmer eingestellt. Diesen Trimmer können Sie auf den gewünschten maximalen Ausgangsstrom einstellen, in dem Sie den Ausgang des Netzteils mit einem Amperemeter kurzschliessen und den gewünschten Ausgangsstrom einstellen.
Wer es wünscht, kann an dieser Stelle natürlich auch ein Poti einsetzen und eine regelbare Strombegrenzung einbauen.
Die Ausgangsspannung des DAC wird auf einen Trimmer mit 10 Kiloohm gegeben. Da der DAC ebenfalls einen Ausgangswiderstand von etwa 10 Kiloohm besitzt, erstreckt sich der Ausgangsspannungsbereich des Wandlers allerdings nur noch über den Bereich von fast 0 bis etwa 1.25 Volt. Diese Beschaltung hat den Vorteil, daß die Offset-Spannung des Wandlers, die kleinste erzielbare Ausgangsspannung, ebenfalls auf die Hälfte reduziert wird. Damit wird die kleinste Ausgangsspannung des Netzteils bei gewünschter Ausgangsspannung von 0 Volt geringer, als wenn die Ausgangsspannung des DAC direkt auf den OP gegeben würde.
Gleichzeitig wird mit dem Poti die maximale Ausgangsspannung der Stromversorgung eingestellt. Bei richtigem Abgleich entspricht dann jede Stufe der Ausgangsspannung genau 100 mV.
Die vom Schleifer des Potis kommende Spannung wird auf den Plus-Eingang des ersten OPs gegeben. Diese Spannung wird mit der Spannung des Spannungsteilers im Ausgang der Stromversorgung (10 KOhm / 330 Ohm) verglichen. Die Ausgangsspannung des OP stellt sich nun auf einen Wert ein, bei dem am Ausgang der Stromversorgung eine Spannung erscheint, die



Mit DAC einstellbares Netzteil

proportional zur Spannung des Wandlers ist und die Differenz der Eingangspannungen des OP 0 Volt beträgt. Erhöht sich die Spannung am Ausgang des DAC, so erhöht sich auch die Ausgangsspannung des OP. Dadurch werden die Transistoren weiter durchgesteuert, die abgegebene Spannung erhöht sich ebenfalls. Über die Rückkopplung vom Spannungsteiler zum Minus-Eingang des OP stellt sich auch hier eine höhere Spannung ein. Dieser Vorgang ist dann beendet, wenn die Differenz zwischen den beiden Eingangsspannungen 0 Volt erreicht.

Der zweite OP übernimmt die Funktion der Stromüberwachung. Gemessen wird dazu der Spannungsabfall über dem Widerstand 0.5 Ohm. Solange kein oder ein geringer Strom fließt, liegt am Plus-Eingang eine höhere Spannung als am Minus-Eingang. Damit ist der Ausgang des OP auf Plus geschaltet. Wird aber durch einen größeren Strom der Spannungsabfall am Meß-Widerstand so groß, daß sich an den Eingängen umgekehrte Verhältnisse einstellen, dann wird der Ausgang Low. Jetzt setzt die Diode im Ausgang des unteren OPs ein. Über diese Diode wird die Basisspannung des ersten Transistors kurzgeschlossen, die Ausgangsspannung sinkt.

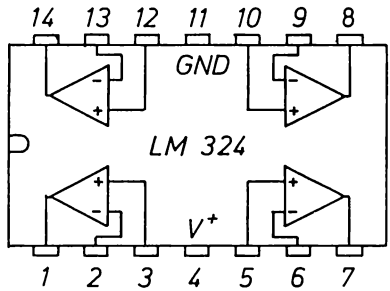
So, nachdem die Funktion der Schaltung geklärt ist, benötigen wir zum Betrieb natürlich auch eine Software. Mit dem nachfolgenden kleinen Programm kann der Wert der gewünschten Spannung direkt über die Tastatur eingegeben werden. Nach dem erfolgten Abgleich können Sie also direkt 10.3 Volt eintippen und sicher sein, daß am Ausgang auch tatsächlich 10.3 Volt anstehen.

```
100 PRINTCHR$(147)
110 C2 = 56576
120 PB = C2 + 1
130 DR = C2 + 3
140 POKE DR,255
```

```
150 POKE PB,0
160 PRINT:PRINT:PRINT"          N E T Z T E I L "
170 PRINT"          =============== "
180 PRINT:PRINT"     EINGESTELLTE SPANNUNG ";SP;" VOLT
190 PRINT:PRINT:PRINT"       F1 = SPANNUNG AENDERN"
200 PRINT:PRINT"       F7 = PROGRAMM BEENDEN"
210 PRINT:PRINT:PRINT "              ?";
220 GET A$:IFA$=""THEN220
230 IFA$=CHR$(133)THENPRINT" F1":GOTO260
240 IFA$=CHR$(136)THEN300
250 GOTO220
260 PRINT:PRINT:PRINT:INPUT"     NEUE SPANNUNG       ";SP
270 SP=INT(SP*10)/10
280 POKEPB,SP*10
290 PRINTCHR$(147):GOTO160
300 PRINTCHR$(147)
310 PRINT:PRINT:PRINT"   DIE  EINGESTELLTE  SPANNUNG  VON "
320 PRINT:PRINT"  ";SP;"VOLT BLEIBT JETZT BIS ZU EINER"
330 PRINT:PRINT"   UMPROGRAMMIERUNG  DES USER-PORTS "
340 PRINT:PRINT"   ERHALTEN. "
350 PRINT:PRINT:PRINT"         A C H T U N G !"
360 PRINT:PRINT:PRINT"   AUS RUN/STOP-RESTORE LOESCHT DIE "
370 PRINT:PRINT"   AUSGANGSSPANNUNG."
```

Mit einem etwas geänderten Programm könnten aber auch andere Anwendungen als die hier gezeigte realisiert werden.
Wenn Sie z.B. die Ausgangsspannung kontinuierlich von 0 Volt bis zum Endwert mit einer FOR NEXT-Schleife hochlaufen lassen, könnten Sie die Ausgangsspannung nutzen, um die Kennlinie von Dioden oder auch Transistoren aufzunehmen. Hier sind Ihrer Phantasie keine Grenzen gesetzt.
Die Verwendung eines Vierfach-OPs vom Typ LM 324 sieht wohl etwas nach Verschwendung aus, da wir nur zwei der eingebauten OPs verwenden. Diese ICs sind aber so preiswert,

daß man sie ruhig auch in solch einer Applikation einsetzen kann. Außerdem wäre mit z.B. einem uA 741 die Ausgansspannung nicht bis auf fast 0 Volt herunterregelbar. Dieses Verhalten ließe sich dann nur mit einer aufwendigeren Schaltung und einer negativen Hilfsspannung erzeugen. Damit würden aber die Kosten für die Schaltung beträchtlich ansteigen.
Unbedingt vorsehen sollten Sie einen ausreichend dimensionierten Kühlkörper für T2. Andernfalls ist die Schaltung nicht kurzschlußfest! Auch darf die Sekundärspannung des Trafos 24 Volt nicht übersteigen, da sonst die OPs zerstört werden können.



6.2.2 Motorregler mit ZN 426

Im Kapitel 4.8 haben wir eine Motorsteuerung kennengelernt, mit der wir einen Gleichstrommotor ein- und ausschalten konnten. Viel interessanter und für Regelschaltungen besser einsetzbar ist aber eine Schaltung, mit der wir den Motor in der Drehzahl regeln können. Eine solche Schaltung wollen wir jetzt kennenlernen.
Wenn Sie den D/A-Wandler und die Motorsteuerstufe mit dem IC TDA 2002 bereits aufgebaut haben, dann benötigen Sie keine weitere Hardware. Der Ausgang des Wandlers wird direkt mit dem Pin 2 des TDA 2002 verbunden.
Wie schon im Kapitel 4.8 erwähnt arbeitet der Verstärker in invertierender Betriebsart. Das bedeutet, daß bei kleiner Eingangsspannung der Motor schnell läuft, bei hoher Eingangsspannung dagegen langsam. Zum Erproben der Schaltung kann das folgende kleine Programm verwendet werden. In diesem Programm kann mit den Funktionstasten F1 und F7 die Geschwindigkeit kontrolliert werden.

```
100 PRINT CHR$(147)
110 C2 = 56576
120 PB = C2+1
130 DR = C2+3
140 POKE DR,255
150 POKE PB,240
160 POKE 650,255
170 FOR I = 1 TO 40: B$ = B$ + " ": NEXT
180 PRINT: PRINT: PRINT"      M O T O R R E G L U N G"
190 PRINT"      ======================="
200 PRINT: PRINT: PRINT:
205 PRINT"  F1                                F7"
210 PRINT"LANGSAM                           SCHNELL"
220 FOR I = 1 TO 40: PRINT CHR$(162);: NEXT
```

```
230 W = 240
240 D = (240-W)
250 POKE PB , W
260 PRINTLEFT$(B$,INT(D/6.1));"↑";RIGHT$(B$,41-(INT(D/6.1)))
270 PRINT CHR$(145); CHR$(145);
280 GET A$: IF A$= "" THEN 280
290 IF A$ = CHR$(133) THEN 310
300 IF A$ = CHR$(136) THEN 330
305 GOTO 280
310 W = W+6: IF W>240 THEN W=240
320 GOTO 240
330 W = W-6: IF W<0 THEN W=0
340 GOTO 240
```

In diesem Programm sind keine Besonderheiten enthalten. Es sollte eigentlich leicht zu verstehen sein. Der POKE in der Zeile 160 schaltet alle Tasten auf Zeichenwiederholung. Dadurch brauchen die Tasten F1 oder F7 nur festgehalten werden und der Bereich wird langsam durchlaufen.
Unser Versuchsaufbau wurde mit einem kleinen Modellbaumotor durchgeführt. Bei anderen Motoren, vor allem solchen mit hoher Stromaufnahme kann es vorkommen, daß der Motor erst bei einer relativ hohen Ausgangsspannung zu drehen beginnt. Dann sollten Sie einen Spannungsteiler vor den Eingang des TDA 2002 schalten. Die Widerstandswerte sind durch Experimentieren sicher schnell ermittelt.
Steigt die durchschnittliche Stromaufnahme des Motors über 100 mA, dann sollten Sie dem Verstärker einen kleinen Kühlkörper spendieren. An sonsten kann es passieren, das die eingebaute Schutzschaltung im Verstärker-IC anspricht. Zerstört wird dabei zwar nichts, die Schaltung arbeitet aber dann nicht mehr wie gewünscht.

7.0 Analog/Digital - der umgekehrte Weg

Überall in unserem täglichen Leben sind wir von analogen Größen umgeben. So stellt die Zimmertemparatur eine analoge Größe dar, aber auch der Luftdruck, die Helligkeit, die Geschwindigkeit eines Autos oder die Lautstärke eines Radios.
Diese Werte mit einem Computer zu verarbeiten und zu speichern gehört zu den interessantesten Problemen der Computertechnik.
Einsatzgebiete für die A/D-Wandlung sind z.B. die Reglung einer Heizungsanlage, Drehzahlmessungen oder auch die Meßtechnik.
Da alle diese analogen Größen unendlich viele Zwischenwerte annehmen können, ist eine Verarbeitung mit einem Computer durch dessen begrenzten Speicherraum immer mit Rundungsfehlern behaftet.
Man muß sich also darüber im Klaren sein, daß der im Computer gespeicherte Wert nur eine Annäherung an den tatsächlichen Analogwert darstellt. In der Praxis ist dieser Fehler aber durch verschiedenen technische Maßnahmen vertretbar gering.

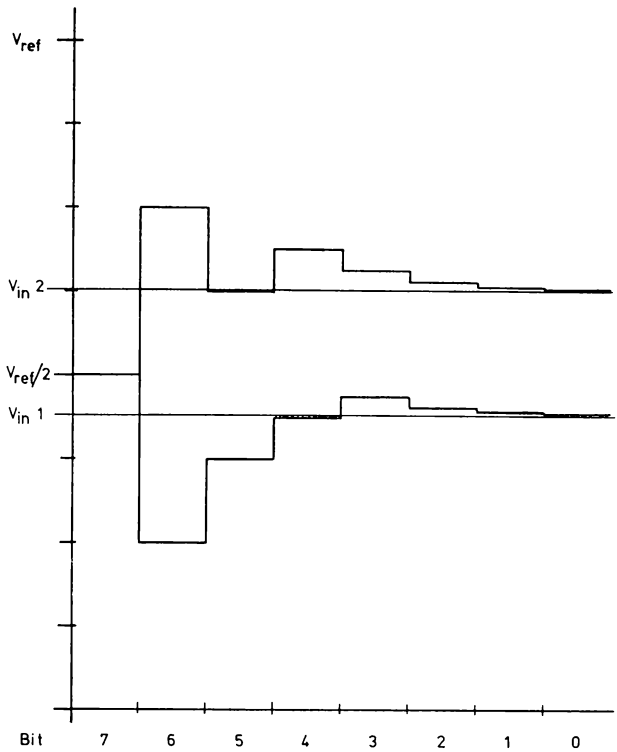
7.1 Methoden der A/D-Wandlung

Wie kann man nun einen analogen Wert, sagen wir z.B. eine Gleichspannung in einen digitalen Wert umwandeln?
Dafür gibt es die unterschiedlichsten Methoden. Die gebräuchlichen Verfahren werden wir in diesem Kapitel kennenlernen, bevor wir uns mit der konkreten Realisation zweier verschiedener A/D-Wandler befassen.
Das einfachste Verfahren, eine Gleichspannung in einen digitalen Wert umzuwandeln, besteht in der Verwendung eines

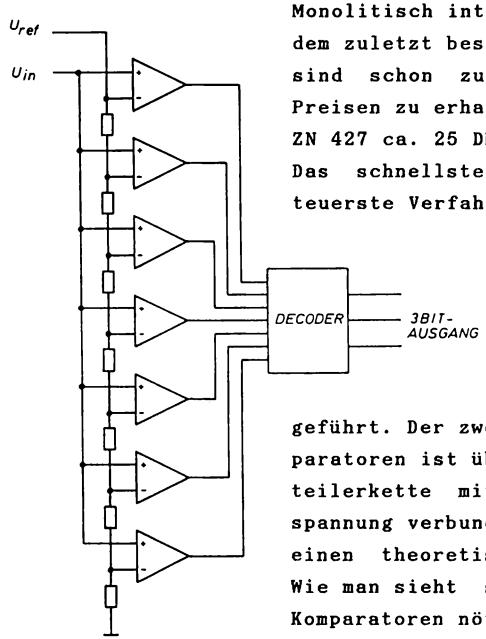
sogenannten VCOs. Ein VCO ist ein spannungsgesteuerter Oszillator (Voltage Controlled Oszillator). Als VCO können beispielsweise ICs wie das ICL 8038 oder das XR 2280 eingesetzt werden. Aber auch diskret, d.h. mit Einzeltransistoren aufgebaute VCOs sind realisierbar.
Wenn die Spannung am VCO-Steuereingang gering ist, dann stellt sich am Ausgang eine niedrige Frquenz ein, bei einer hohen Spannung erhalten wir eine hohe Frequenz. Wenn nun der Computer durch ein geeignetes Programm in der Lage ist, für einen bestimmten Zeitraum Impulse zu zählen, dann stellt die Anzahl der Impulse ein Maß für die angelegte Gleichspannung dar.
Dieses Verfahren der A/D-Wandlung hat, obwohl es sich so schön einfach anhört, einige wesentliche Nachteile. Da ist zum einen die sehr lange Meßzeit. Je nach Dimensionierung des VCO und geforderter Meßgenauigkeit sollten für mindestens 0,1 Sekunde die Impulse gezählt werden. Zur Erzielung einer größeren Genauigkeit muß die Meßzeit entsprechend verlängert werden. Auch läßt die Linearität der VCOs im Regelfall gerade im Bereich kleiner Eingangsspannungen zu wünschen übrig. Das macht einen umständlichen Kalibriervorgang nötig, der die Anzahl der Bauteile und somit die Kosten erhöht.
Eine andere, häufig in Digitalvoltmetern eingesetzte Methode der A/D-Wandlung stellt das sogenannte DUAL-SLOPE-Verfahren dar. Das Funktionsprinzip ist etwas komplizierter als der reine Zähler-Wandler, aber auch dies Prinzip basiert auf einer Zählmethode. Zuerst wird mit der Eingangsspannung über eine Konstantstromquelle ein Kondensator aufgeladen.
Der Aufladevorgang dauert immer eine konstante Zeit. Zur Zeiterzeugung wird ein Oszillator eingesetzt, der auf einer vergleichsweise hohen Frequenz läuft, und dessen Signal durch Flip-Flops heruntergeteilt wird. Durch die konstante Ladezeit erzeugen unterschiedliche Eingangsspannungen zu

dieser proportionale Kondensatorladungen.
Nach dem Aufladen wird der Kondensator mit einem von einer Referenzspannung erzeugten Konstantstrom entladen. Die Dauer der Entladezeit stellt jetzt ein Maß für die Eingangsspannung dar, da der Kondensator bei kleinen Spannungen schnell entladen, bei hohen Spannungen langsamer entladen wird.
Der Vorteil dieses Verfahrens: Die Frequenz des Oszillators muß nur in der Zeit einer Messung konstant bleiben. Ändert sich die Frequenz in einem größeren Zeitraum langsam, so leidet die Meßgenauigkeit nicht. Auch an den Meßkondensator werden keine besonderen Anforderungen gestellt.
Allerdings ist auch bei diesem Meßprinzip die Meßrate, also die Anzahl der Messungen pro Sekunde sehr gering. Bei verschiedenen handelsüblichen ICs, welche mit dem beschriebenen Prinzip arbeiten, betragen die Meßraten etwa 3 bis ca. 100 Messungen pro Sekunde. Später werden wir eine vollständige Schaltung mit einem solchen Digitalvoltmeter-IC beschreiben und programmieren.
Die bisher beschriebenen Prinzipien der A/D-Wandlung sind aber alle nicht besonders geeignet, die Daten an einen Computer zu übergeben. Darum haben sich die Entwicklungsingeneure andere brauchbarere Methoden ersonnen.
Ein bekanntes, in der Computertechnik gerne eingesetztes Verfahren ist die sogenannte sukzessive Approximation. Hinter diesem Wortmonster verbirgt sich eine gleichermaßen schnelle wie genaue und einfache Wandlungsmethode.
Im Detail funktioniert die Sache so:
Das Eingangssignal wird auf einen Komparator, einen Spannungsvergleicher gegeben. An den anderen Eingang des Komparators wird nun eine Spannung gelegt, die der Hälfte der maximalen Eingangsspannung entspricht. Ändert sich das Ausgangssignal des Komparators, dann ist die Eingangsspannung kleiner als die Hälfte des Maximalwertes, sonst ist,

sie größer. Ist die Eingangsspannung nun geringer, so wird die Vergleichsspannung auf die Hälfte reduziert, ansonsten um die Hälfte erhöht. Danach wird wieder entschieden, ob als nächstes die Vergleichsspannung erhöht oder erniedrigt werden muß. Durch den Vergleich nähert sich die Vergleichs- oder Referenzspannung schrittweise an den Eingangswert. Die Zeichnung soll den Vorgang noch einmal für zwei verschiedene Eingangsspannungen verdeutlichen.



Nach diesem Prinzip gebaute Wandler können innerhalb von nur ca. 10 Microsekunden eine Spannung in einen 8-Bit-Digitalwert wandeln. Das bedeutet, daß in einer Sekunde 100000 Messungen vorgenommen werden können. Der digitalisierte Wert steht direkt in paralleler Form zur Verfügung und kann zum Beispiel über den User-Port eingelesen werden.



Monolitisch integrierte Wandler nach dem zuletzt beschriebenen Verfahren sind schon zu relativ günstigen Preisen zu erhalten. So kostet der ZN 427 ca. 25 DM.

Das schnellste, allerdings auch teuerste Verfahren ist die sogenannte Parallelumsetzung. Bei dieser Methode wird das Eingangssignal auf eine ganze Kette von Komparatoren geführt. Der zweite Eingang der Komparatoren ist über eine Spannungsteilerkette mit einer Referenzspannung verbunden. Das Bild zeigt einen theoretischen 3-Bit-Wandler. Wie man sieht sind dafür schon 7 Komparatoren nötig.

Soll die Auflösung z.B. 8 Bit betragen, so sind insgesamt 2↑8-1 = 255 Komparatoren nötig. Damit erklärt sich auch der sehr hohe Preis für diese Wandler. Da das digitale Ergebnis aber nicht von irgendwelchen Zählern oder sonstigen zeitbestimmenden Gliedern abhängt, kann man das Ergebnis bei einigen Typen schon nach 50 Nanosekunden erwarten. Bei diesen Geschwindigkeiten lassen sich sogar Fernsehbilder digitalisieren und im Computer speichern. Allerdings liegen die Kosten für diese Wandler deutlich über 100 DM, kein sehr billiger Spaß.

7.2 Der eingebaute A/D-Wandler

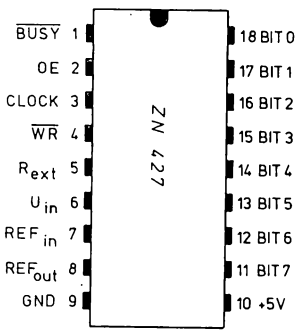
Die im C64 eingebauten zwei A/D-Wandler haben ein etwas eigenwilliges Funktionsprinzip. Es stellt eine Abart des Zählerprinzips dar.
Bei diesen A/D-Wandlern wird über das angeschlossene Paddle ein Kondensator für 0.25 Millisekunden aufgeladen. Dabei wird die Spannung am Kondensator von einem Komparator überprüft. Sobald die Spannung größer als eine im SID erzeugte Referenzspannung wird, stoppt ein Zähler. Dieser Zähler wird vom Systemtakt angesteuert.
Ist der Widerstandswert des Paddle klein, so wird der Kondensator schnell geladen, der Zähler wird entsprechend früh gestoppt. Bei größeren Widerstandswerten ergibt sich ein höherer Zählerstand.
Wird der Widerstand zu groß, also etwa über 200 KOhm, dann wird der Zähler in der Meßzeit nicht angehalten und erreicht so seinen Endwert von 255. Bei zu kleinen Widerstandswerten stoppt der Zähler, bevor ein Impuls gezählt werden konnte. Dann steht im A/D-Register der Wert 0.
Nach Ablauf der Meßzeit wird der Kondensator schlagartig über den A/D-Eingang entladen. Nach weiteren 0.25 Millisekunden beginnt darauf ein neuer Meßzyklus.
Damit ist auch die maximale Wandlungsrate festgelegt. Sie beträgt 2000 Messungen pro Sekunde.
Leider läßt sich mit den A/D-Wandlern direkt keine Spannung messen. Dazu ist die Meßgenauigkeit auch nicht ausreichend.
Alle Versuche, den Widerstand z.B. durch eine spannungsgesteuerte Konstantstromquelle zu ersetzen, scheiterten an der extremen Unlinearität des Wandlers.

7.3 Der ZN 427E

In diesem Kapitel wollen wir einen monolitischen A/D-Wandler untersuchen und einige Experimente damit ausführen.
Der ZN 427E von Ferranti zeichnet sich durch Qualität, günstigen Preis und nicht zu letzt duch sehr gute Verfügbarkeit aus. Die wichtigsten technischen Daten dieses 8-Bit-Wandlers sind:

Einfacher Betrieb in Microcomputersystemen,
Voll TTL und CMOS-Kombatibel
Sukzessive Approximation, kurze Wandlungszeit
Integrierte temparaturstabile Referenzspannung
Bis auf einige Widerstände keine zusätzliche Beschaltung

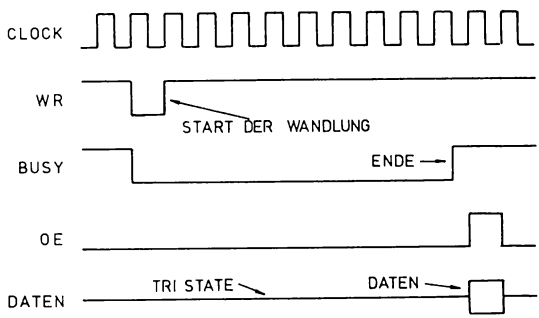
Ein kleines Handicap soll aber auch nicht verschwiegen werden, das IC benötigt eine negative Versorgungsspannung. Dafür werden wir uns aber eine Trickschaltung einfallen lassen.



Die Anschlüsse dieses 18-poligen ICs haben folgende Bezeichnungen:
Der Pin 6 ist der Eingang für das analoge Signal. An den Pins 11 bis 18 kann der digitalisierte Wert der Eingangsspannung abgenommen werden.
Die benötigte Referenzspannung ist am Pin 8 verfügbar. Sie beträgt 2.56 Volt. Wenn dieser Anschluß mit Pin 7 verbunden wird, so

beträgt der Meßbereich des Wandlers ohne sonstige Vorkehrungen ebenfalls 2.56 Volt. Das Verbinden der beiden Anschlüsse ist die einfachste Art, die benötigte Referenzspannung zu erhalten.
Die erzeugte Spannung ist sehr stabil und Temparatureinflüssen nur in sehr geringem Maße unterworfen. Laut Datenblatt beträgt der Temparaturkoeffizient typisch 50 ppm/Grad Celsius.
Die Bezeichnung ppm bedeutet übrigens Part Per Million oder Teile pro Millionen. Die Referenzspannung ändert sich also bei einer Temparaturänderung von 1 Grad Celsius um nur 50 millionstel!

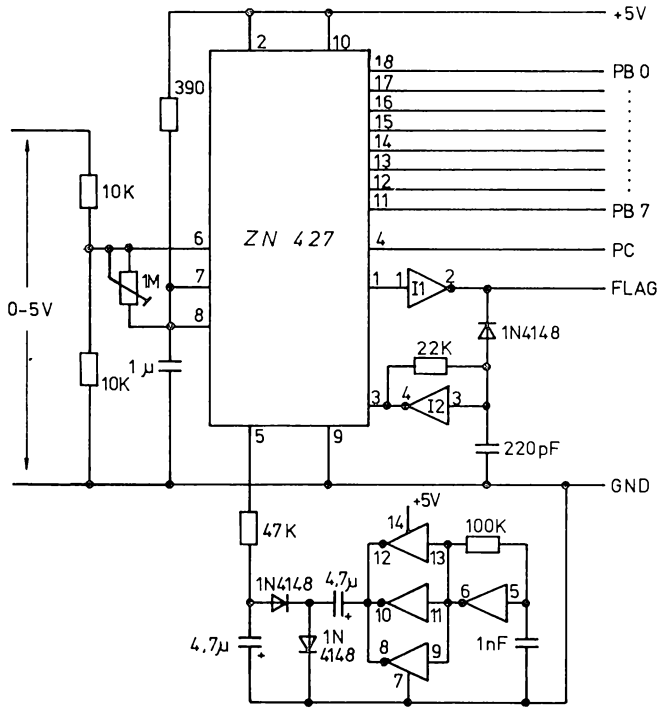
TIMING des ZN427



Die Pins 1, 2, 3 und 4 stellen die Steuersignale zum Rechner dar. Der Pin 4 mit der Bezeichnug -WR ist der Eingang, über den eine Wandlung gestartet wird. Die Wandlung beginnt mit der positiven Flanke eines negativen Impulses.
Nachdem Pin 4 auf Masse gelegt wird, wird sofort der Pin 1 (Busy) auf Masse gelegt. Das signalisiert dem angeschlossenen Rechner, daß momentan keine gültigen Daten an den Ausgängen 11 bis 18 anliegen.

Die Wandlung benötigt 9 Taktimpulse. Diese Taktimpulse werden dem IC am Pin 3 (Clk) zugeführt. Dabei ist es unerheblich, ob hier genau 9 Impulse erzeugt werden, oder ob eine kontinuierliche Rechteckspannung anliegt.
Nach den 9 Taktimpulsen an Clk wird der Pin 1 (Busy) wieder High, dem Prozessor wird anzeigen, daß ein Wert von den Pins 11 bis 18 'abgeholt' werden kann. Um den Wert zu lesen, ist es notwendig, den Pin 2 (RD) auf ein High-Potential zu legen. Je nach Verwendung und zusätzlicher Beschaltung kann dieser Pin permanent an Plus liegen oder über einen Dekoder angewählt werden. In diesem Beispiel werden wir von der ersten Methode Gebrauch machen.
Bleibt noch der Anschluß Rext Pin 5. An diesen Anschluß muß eine negative Spannung von -3 bis -15 Volt(!) angeschlossen werden.
Den fachlich versierten Lesern sei an dieser Stelle kurz der Zweck dieser negativen Spannung erklärt.
Die analoge Eingangsstufe des ZN 427 ist als Differenz-eingang mit zwei Transistoren aufgebaut. Die Emitter der beiden Transistoren sind miteinander verbunden und auf den Pin 5 gelegt. Der positive Eingang ist als Analogeingang auf den Pin 6 gelegt während der invertierende Eingang mit dem D/A-Wandler im IC verbunden ist. Um nun Spannungen bis 0 Volt zu messen, ist es notwendig, die Emitter auf ein mindestens .7 Volt negativeres Potential als die Basen zu legen. Diese negative Vorspannung der Emitter wird über den Widerstand Rext und der negativen Versorgungsspannung erzeugt.
Aber genug der trockenen Theorie!
Machen wir den Lötkolben heiß und fangen wir an.
Die abgebildetet Schaltung ist zwar sehr einfach, leistet aber hervorragendes. In allen Fällen, wo mit einem C64 eine analoge Größe verarbeitet wrden soll und die Genauigkeit von 8 Bit ausreicht, ist diese Schaltung vermutlich das Non Plus

Ultra. Mit nur zwei ICs wurde ein vollständiger Wandler aufgebaut, der rein hardwaremäßig arbeitet, und dessen Werte mit einem einzigen PEEK in den C64 geholt werden können. Zum Betrieb ist keine spezielle Software nötig!



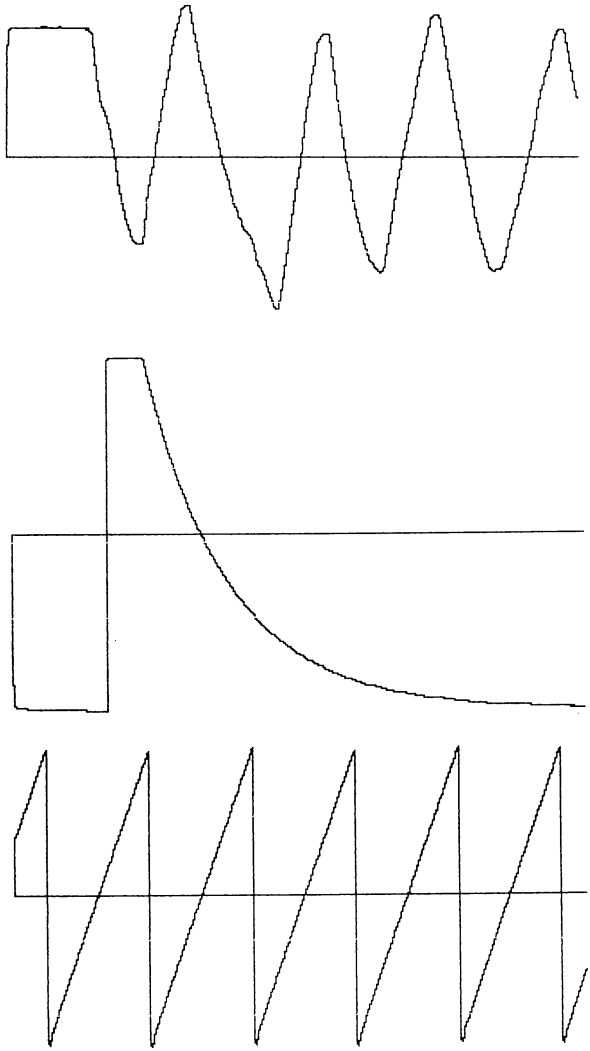
Die beiden verwendeten ICs sind der ZN 427 und ein IC mit 6 CMOS Invertern mit Schmitt-Trigger-Eigenschaften, die Typenbezeichnung ist 4584.

Anders als bei TTL-ICs haben die bei CMOS-ICs vor der Bezeichnung aufgeführten Buchstaben keine Bedeutung. So ist in der Schaltung ein HEF 4584 ebensogut einsetzbar, wie ein MC 4584. Auch andere Buchstabenkombinationen sind denkbar,

jeder Hersteller kennzeichnet seine ICs mit seinem eigenen Kürzel.
Aber kommen wir zur Funktion der Schaltung.
Betrieben wird die Schaltung am User-Port des C 64. Die User-Portleitungen PB0 - PB7 sind mit den Digitalausgängen Pin 11 bis 18 des A/D-Wandlers verbunden. Da der Pin 2 des Wandlers auf 5 Volt liegt, liegen die Werte des Wandlers immer am Port B der CIA 2 an.
Wird dieser Port gelesen, so wird automatisch am Anschluß -PC ein negativer Impuls von ca. 1 Microsekunde erzeugt. Da aber dieser Anschluß mit dem Pin 4 des Wandlers verbunden ist, wird sofort mit dem Lesen des Wertes von Port B ein neuer Meßvorgang eingeleitet.
Wie schon erwähnt wird nach einem -WR-Impuls der Anschluß -BUSY des Wandlers nach Masse gelegt. Dieses Signal wird über den Inverter I1 auf den -Flag-Eingang geführt, zusätzlich wird aber auch der mit dem Inverter I2 aufgebaute Oszillator gestartet.
Dieser Oszillator (bei genauem Hinsehen sind es in dieser Schaltung gleich zwei solche Oszillatoren) arbeitet nach folgendem Prinzip. Nach dem Einschalten ist der Kondensator zwischen Eingang und Masse entladen. Damit liegt der Eingang des Inverters auf Masse und der Ausgang der Stufe ist High. Über den Widerstand zwischen Aus- und Eingang wird aber der Kondensator aufgeladen. Dadurch wird nach einiger Zeit der Pegel am Eingang (also Über dem Kondensator) so hoch, daß der Eingang ein High erkennt. Schlagartig wird der Ausgang der Stufe Low, der Kondensator wird jetzt über den Widerstand entladen. Diese Entladung geschieht aber nur so lange, bis der Eingang wieder ein Low erkennt. Dann schaltet der Ausgang sofort auf High und der beschriebene Vorgang beginnt von vorn.
Wir können also mit der beschriebenen Schaltung ein Rechtecksignal erzeugen. Die Frage ist, wie jetzt die

beschriebenen 9 Taktimpule erzeugt werden.
Nun, des Rätsels Lösung ist die Diode D1. Sobald der Pin BUSY des Wandlers Low ist, wird der Ausgang von I1 High, die Diode D1 sperrt und der Oszillator kann schwingen.
Wenn der BUSY-Pin dagegen High ist, haben wir am Ausgang von I1 ein Low und über die Diode ist auch der Eingang von I2 auf Low. Damit wird der Kondensator immer entladen, es kann keine Rechteckschwingung erzeugt werden.
In der angegebenen Dimensionierung beträgt die Periodendauer des Start-Stop-Oszillators etwa 5 Microsekunden, eine komplette Wandlung ist also nach etwa 50 Microsekunden abgeschlossen.
Der mit dem Inverter I3 aufgebaute Oszillator verfügt nicht über die Möglichkeit, ihn zu stoppen. Er ist als frei schwingender Oszillator aufgebaut. Zusammen mit den verbleibenden drei Invertern wird hiermit die negative Hilfsspannung erzeugt. Die Erzeugung dieser negativen Spannung ist etwas trickreich.
Während am Ausgang der drei Inverter ein Signal anliegt, daß zwischen 0 und +5 Volt schwankt, ist nach dem Kondensator eine echte Wechselspannung zwischen +2.5 Volt und -2.5 Volt zu messen. Diese Wechselspannung wird über den mit den Dioden D2 und D3 aufgebauten Spannungsverdoppler gleichgerichtet. So entsteht eine negative Spannung von ca. 3.5 - 4 Volt.
Eine ausführliche Erläuterung der Schaltung würde den Rahmen diese Buchs sprengen. Interessenten sei in diesem Fall die verfügbare Literatur über grundsätzliche Halbleiter-schaltungen empfohlen.
Aus der Beschreibung der Hardware ergibt sich allerdings ein interessanter und nicht zu vergessener Aspekt.
Sobald ein Wert vom User-Port gelesen wird, startet eine neue Konversion. Das bedeutet, daß der gelesenen Wert nicht den tatsächlich zur Zeit anliegenden Wert ergibt. Statt

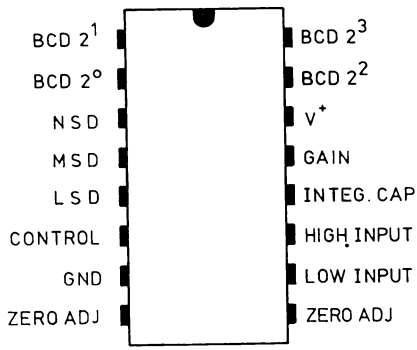
dessen wird ein Wert geliefert, der zur Zeit des vorhergegangenen PEEKens des Port B anlag. Beim ersten Zugriff wird somit überhaupt kein vernünftiger Wert geliefert.
Um also den zu einem Zeitpunkt tatsächlichen Wert zu ermitteln, sind zwei PEEKs auf Port B des User-Ports nötigt. Der mit dem ersten PEEK geholte Wert wird dann verworfen und der Wert des zweiten PEEK hat Gültigkeit für die Zeit des ersten PEEK.
In der Praxis müssen Sie selbst entscheiden, ob Sie jeden Wert benutzen, oder ob Sie vom Prinzip der doppelten Messung Gebrauch machen.
Zum Abschuß dieses Kapitels sind auf der nächsten Seite einige Hardcopys abgedruckt. Diese Hardcopys entstanden mit Hilfe der DATA BECKER SUPERGRAPHIK. Dargestellt sind von oben nach unten eine Aufzeichnung eines angeschlossenen Potis, das über die Meßzeit einige Male verstellt wurde. Als zweites ist eine Entladekurve eines Kondensators aufgezeichnet und als letztes wurde die Ablenkspannung eines Oszillographen auf den A/D-Wandlereingang gegeben.
Die Graphiken wurden erstellt mit einem kleinen Programm, in dem die Werte des Wandlers mit PEEK 56577 eingelesen und nach einer Umrechnung mit dem PLOT-Befehl auf dem Bildschirm dargestellt wurden.
Alles in allem ergibt sich mit diesem IC und der gezeigten Schaltung für den Hobbyisten eine Unmenge von verschiedenen Anwendungen. In Maschinensprache programmiert ist sogar eine Art Oszilloskop mit Speicherung machbar.
Also, frisch ans Werk. Es gibt viel zu tun, packen Sie's an.



7.4 Der CA 3162

Das IC CA 3162 von RCA enthält alle für ein Digitalvoltmeter benötigten Stufen. Das verwendete Meßverfahren ist das schon erläuterte Zwei-Flanken-Prinzip, auch Dual-Slope Verfahren genannt.
Entwickelt wurde dies IC für den Einsatz in Low-Cost-Digitalvoltmetern (DVM). Es hat einen Meßbereich von - 99 mV bis 999 mV. Dabei benötigt das IC nur eine Versorgungsspannung von 5 Volt! Dieser Punkt war bei der Wahl des ICs mit ausschlaggebend, da verschiedene andere DVM-ICs immer auch eine negative Versorgung benötigen.

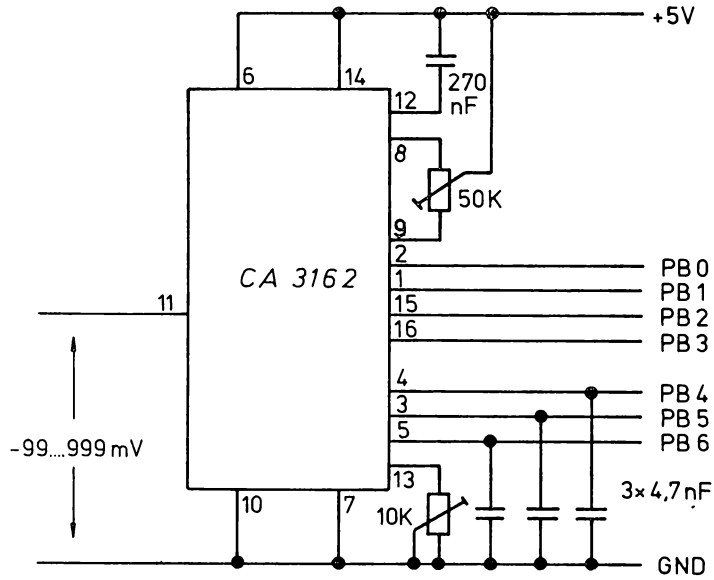
Für den Betrieb sind außer einem Eingangsspannungsteiler nur noch zwei Potis und ein Kondensator nötigt. In einem 'richtigen' DVM wird noch ein Dekoder zur Ansteuerung der Sieben-Segment-Anzeigen benötigt, dieser Dekoder wird bei unserer Anwendung aber überflüssig und durch Software ersetzt.



Anders als beim ZN 427 werden die Daten aber nicht im binären Format ausgegeben, die Ausgangsstufe des CA 3162 ist ein gemultiplexter BCD-Ausgang. Gemultiplext bedeutet, daß an dem Ausgang die Information zeitlich nacheinander erscheint. Zuerst liegt an den Pins 1, 2, 15 und 16 die Information der höchsten Anzeigestelle an. Das wird durch einen Low-Impuls am Pin 4 signalisiert. Nach ca. 5 Millisekunden wird Pin 4 wieder High, dafür wird nun Pin 5 Low. Gleichzeitig wechselt die

Information auf den BCD-Ausgängen. Jetzt liegt die niederwertige Anzeigenstelle an den Ausgängen. Nach wiederum 5 Millisekunden kommt dann die Information der mittleren Anzeigestelle. Für diese Stelle signalisiert der Pin 3 durch ein Low die Gültigkeit der Daten.
Dadurch dieses Anzeigeverfahren wird die Software etwas aufwendiger, als dies im vorigen Kapitel nötig war. Leider fehlt auch ein Ausgang, der das Ende einer jeden Messung anzeigt. Da dies IC aber nicht für den Einsatz in Microcomputersystemen entwickelt wurde, müssen wir uns eine kleine Software entwickeln, die dennoch einen Einsatz ermöglicht. Daß diese Software zum Teil in Maschinensprache geschrieben ist, sollte Sie nicht abschrecken. Das Programm besteht aus nur wenigen Bytes und die Funktion werden wir für die 'Nur'-Basic-Programmierer ausführlich erklären.
Aber kommen wir nach so viel Vorrede doch erst einmal zum praktischen Teil, der Hardware.
Der für das Dual-Slope-Prinzip nötige Integrationskondensator wird zwischen + 5 V und dem Pin 12 angeschlossen.
Ein an den Pins 8 und 9 angeschlossenes Poti wird zur Justage des Messwertes auf 0 Volt bei kurzgeschlossenen Eingängen benötigt, während das zweite Poti zwischen Pin 12 und Masse zur Einstellung des Endwertes dient.
Die Eingangsspannung wird an Pin 11 zugeführt.
Der Anschluß 6 des IC bestimmt die Anzeigerate des ICs. Liegt dieser Anschluß an Masse, so beträgt die Anzeigerate 4 Hertz, bei +5V erhöht sie sich auf ca 96 Hertz. Wird der Anschluß auf eine Spannung von 1.2 - 2.8 Volt gelegt, so wird die Anzeige 'eingefroren', der zuletzt gemessene Wert wird jetzt immer angezeigt. Im Hinblick auf eine möglichst rasche Umsetzzeit legen wir diesen Anschluß auf +5V.

Ein mit diesem IC aufgebautes 'nacktes' Digitalvoltmeter besteht aus nur 7 Bauteilen.



Die drei Kondensatoren C1, C2 und C3 werden benötigt, um durch das Multiplexen entstandene Störimpulse zu beseitigen. Werden die drei Kondensatoren weggelassen, so ergibt sich eine stark variierende Anzeige.

Wenn Sie die Hardware aufgebaut haben, kontrollieren Sie vorsichtshalber noch einmal Ihren gesamten Aufbau. Danach stecken Sie das fertige Modul auf den User-Port. Jetzt können Sie den Rechner einschalten und die Software laden. Nach 'RUN' sollte in der linken oberen Ecke eine dreistellige Zahl erscheinen. Diese Zahl ist der vom DVM gemessene Wert. Und jetzt geschieht das Wunderbare. Obwohl Sie an dem Eingang des DVM gar keine Spannung anliegen

haben, wird doch ein Wert angezeigt. Die Lösung dieses Rätsels ist recht einfach. Zum einen haben wir das Voltmeter noch nicht auf den Nullpunkt eingestellt. Zum zweiten ist der Eingang des ICs extrem hochohmig. Er beträgt typisch 100 MOhm! Dadurch können sich statische Ladungen an dem Eingang aufbauen, die dann als Wert angezeigt werden. Darum wollen wir zuerst den 0-Punkt einstellen. Dazu wird der Eingang des IC einfach mit Masse verbunden. Jetzt kann mit dem Poti P1 die Anzeige auf 0 Volt eingestellt werden. Damit ist das erste Poti eingestellt.

Das zweite Poti ist für die Messgenauigkeit zuständig, oder anders ausgedrückt, mit diesem Poti wird bei einer Eingangsspannung von 500 Millivolt auch die Anzeige auf 500 eingestellt. Es stellt sich jetzt nur die Frage, woher wir eine genaue und bekannte Spannung nehmen können. Im einfachsten Fall kann der Abgleich mit einen zweiten Digitalvoltmeter vorgenommen werden. Wenn Sie oder einer Ihrer Bekannten nicht über ein solches Gerät verfügt, dann kommen noch zwei andere Möglichkeiten in Betracht.

Die erste Möglichkeit ist die Anschaffung eines speziellen ICs. In Frage kommen z.B. die ICs LM0070-1, eine 10 V Referenz oder das LM336, eine 2.5 V Referenzspannungsquelle. Damit und mit zwei mindestens 1%-igen Widerständen kann eine sehr genaue Justage erfolgen.

Sind diese ICs nicht zu beschaffen oder zu teuer (das LM0070-1 kostet immerhin ca. 25 - 30 DM, also mehr als die ganze DVM-Schaltung), dann gibt es eine preiswerte Alternative. Die in Taschenrechnern und Armbanduhren eingesetzten Quecksilber-Batterien liefern eine ziemlich genaue Spannung von 1.37 Volt. Diese Spannung ist auf ca. 3% genau. Mit dem im Schaltplan gestrichelt eingezeichneten Spannungsteiler aus drei 10K-Widerständen kann die Anzeige dann auf 0.913 Volt eingestellt werden. Auf diese Weise lassen sich Genauigkeiten von immerhin 5% erzielen.

Ansonsten weist die Schaltung keine Besonderheiten auf.
Doch kommen wir zur benötigten Software.
Wie bereits erwähnt, ist eine Programmierung in Maschinensprache nicht zu umgehen. In Basic programmiert ergeben sich doch Schwierigkeiten, die einzelnen Digits über den User-Port einzulesen. Bei etwas unglücklicher Programmierung kann das so weit gehen, daß bedingt durch die Laufzeit des verwendeten Programms eine einzelne Stelle überhaupt nicht oder nur alle 30 Sekunden gemessen wird. Das ist aber in der Praxis nicht verwertbar. Aus diesem Grund enthält das Programm zum DVM eine kleine Routine in Maschinensprache, die nach dem Start des Programms in den Cassettenpuffer gePOKEd wird. Wird diese Routine mit SYS angesprungen, so liefert sie in drei Speicherzellen in der Zero-Page die drei Digits oder Stellen. Das Basic-Programm liest diese Werte mit PEEK und zeigt sie an.

```
033C                    .OPT P1,OO
033C                    *=   $033C
033C          CIA2      =    $DD00
033C          PB        =    CIA2 + 1
033C          DR        =    CIA2 + 3
033C          MERK      =    $FF
033C 78                 SEI
033D A9 10              LDA  #%00010000
033F 85 FF              STA  MERK
0341 A2 F9              LDX  #$F9
0343 AD 01 DD LOOP      LDA  PB
0346 A8                 TAY
0347 49 FF              EOR  #$FF
0349 29 F0              AND  #%11110000
034B C5 FF              CMP  MERK
034D D0 F4              BNE  LOOP
034F 98                 TYA
```

```
0350 29 0F          AND  #%00001111
0352 95 00          STA  0,X
0354 E8             INX
0355 A5 FF          LDA  MERK
0357 0A             ASL
0358 85 FF          STA  MERK
035A 10 E7          BPL  LOOP
035C 58             CLI
035D 60             RTS
```

Insgesamt werden 4 Speicherzellen in der Zeropage durch das Programm belegt. Es sind dies die Adresses 249 bis 251 für die drei Stellen sowie die Speicherzelle 255. Diese Adressen werden vom Betriebssystem des C64 nur im Zusammenhang mit dem RS-232 Modul benutzt. Der gleichzeitige Betrieb von DVM und diesem Modul ist aber schon physikalisch unmöglich. Es kann ja nur ein Modul zur Zeit auf den User-Port gesteckt werden. Damit stehen diese Adressen zur freien Benutzung.
Wie wird nun im Einzelnen entschieden, welche Anzeigestelle gerade eingelesen wird?
Durch die Beschaltung liegt die Stelleninformation auf den drei Portbits PB4 bis PB6. Ist PB4 Low, so liegt an PB0 bis PB3 die Information des MSD (Most Significant Digit), der höchstwertigen Stelle an. PB5 auf Low signalisiert NSD, die mittlere Anzeigestelle und PB6 die letzte Anzeigestelle (LSD).
Nachdem der Inhalt des Ports B in den Akku geladen wurde, wird dieser Wert in das Y-Register geschrieben und dort aufbewahrt.
Anschließend wird der Akku mit dem EOR invertiert und mit AND %11110000 verknüpft. Jetzt ist im Akku entsprechend der angezeigten Stelle entweder Bit 4, 5 oder 6 gesetzt. Alle anderen Bits sind gelöscht. Jetzt wird der Akku mit dem Inhalt der Speicherzelle MERK ($FF) verglichen. Wird keine

Übereinstimmung festgestellt, so wird ein neuer Wert vom User-Port geholt. Diese Vorgänge wiederholen sich, bis eine Übereinstimmung von Akku und MERK festgestellt werden.
Jetzt wird der ursprüngliche Wert aus dem Y-Register wieder in den Akku geholt. Durch den Befehl AND %00001111 wird die Stelleninformation ausgeblendet. Damit bleibt der Meßwert über. Anschließend wird der Akkuinhalt in der Zero-Page abgespeichert. Dabei gibt das X-Register an, an welche Speicherzelle der Wert transportiert werden soll.
Der Befehl ASL schiebt den Inhalt der Adresse MERK um eine Stelle nach links, d.h. war der ursprüngliche Wert %00010000, so enthält sie nach ASL den Wert %00100000, das gesetzte Bit ist gewandert. Mit diesem Wert wird jetzt auf die nächste anzuzeigende Stelle gewartet.
Nach drei Zeichen wird das Bit in MERK an die höchste Stelle geschoben. Das ist für unser Programm das Zeichen, in das Basic-Programm zurückzukehren.

```
100 FOR I = 828 TO 862
110 READ X : POKE I,X : S=S+X : NEXT
120 IF S <> 5018 THEN PRINT "FEHLER IN DATAS !!" : END
130 PRINT CHR$(147)
140 SYS 828
150 FORI = 0 TO 2
160 GOSUB 200
170 Z$ = Z$ + RIGHT$(A$,1)
180 NEXT
190 PRINT CHR$(14);Z$ : Z$ = "" : GOTO 140
200 A = PEEK (249+I)
210 IF A = 10 THEN A$ = "-" : RETURN
220 IF A = 11 THEN A$ = "↑" : RETURN
230 A$ = STR$(A)
240 RETURN
250 DATA 120,169, 16,133,255,162,249,173,  1,221,168, 73
```

```
260 DATA 255, 41,240,197,255,208,244,152, 41, 15,149,  0
270 DATA 232,165,255, 10,133,255, 16,231, 88, 96
```

Das Basicprogramm enthält keine Besonderheiten. Die Zahlen werden aus den Speicherzellen 249 - 251 heausgePEEKd und in einen String verwandelt. Diese Wandlung ist nötig, sonst könnte die Überlaufanzeige '↑↑↑' und die Unterlaufanzeige '---' nicht realisiert werden.
Das Basic-Programm ist absichtlich etwas einfach gehalten. Ein nacktes DVM ist zwar für viele Anwendungen einsetzbar, wir haben aber ein 'richtiges' DVM mit automatischer Meßbereichswahl für Gleichspannungen bis 100 V entwickelt. Die Schaltung und eine komfortable Software finden Sie im Kapitel 9.

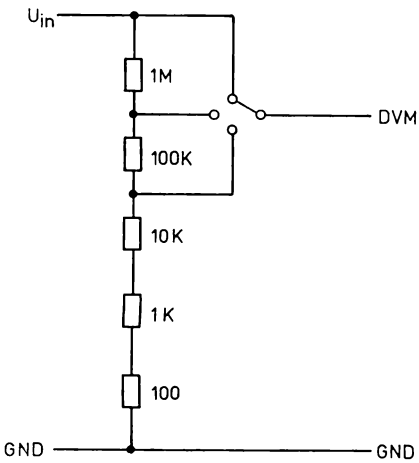
7.4.1 Spannungsmessung

So, nachdem wir jetzt ein Digitalvoltmeter-IC mit Erfolg an den User-Port des C64 angeschlossen haben, wollen wir uns einmal überlegen, was damit alles zu messen ist.
Die naheliegenste und auch im Namen schon enthaltene Anwendung ist natürlich die Spannungsmessung. Aber auch Strom- und Temparaturmessungen sind ohne großen Aufwand realisierbar.
Darüberhinaus sind über geeignete Wandler aber auch andere Größen messbar. So könnten Druck, Feuchtigkeit oder Lichtstärken ohne große Probleme gemessen werden.
Wir werden jetzt nacheinander die drei zuerstgenannten Meßprinzipien kennenlernen.
Betrachten wir zunächst die Spannungsmessung.
Mit dem im vorigen Kapitel beschriebenen DVM-Modul können ohne zusätzliche Beschaltung nur Spannungen zuwischen -99 und +999 Millivolt gemessen werden. Soll der Messbereich vergrößert werden, so ist ein Spannungsteiler vorzuschalten. Die Formel, nach der ein einfacher Spannungsteiler berechnet wird, lautet:

$$\frac{U1}{U2} = \frac{R1}{R2}$$

Im Interesse einer möglichst großen Meßgenauigkeit sollten die verwendeten Spannungsteilerwiderstände eine kleine Toleranz haben. Gut erhältlich sind mittlerweile 1%ige Widerstände, von verschiedenen Anbietern sind aber auch 0.1%ige Meßwiderstände zu bekommen. Werden diese genaueren Widerstände eingesetzt, dann sollte unbedingt der Abgleich des Meßbereichs mit einem genauen Vergleichsvoltmeter vorgenommen werden.

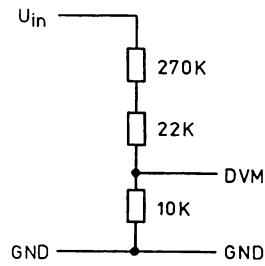
Ein genauer Spannungsteiler für drei Meßbereiche sieht folgendermaßen aus:



Der Spannungsteiler ermöglicht mit unserem DVM die drei Eingangsbereiche 1 Volt, 10 Volt und 100 Volt. Dabei beträgt der Eingangswiderstand in allen Bereichen 1.111 MOhm. Möglicherweise werden Sie sich über die zwei unteren Spannungsteiler-Widerständ e 1K und 100 Ohm wundern. Wenn Sie aber einen Spannungsteiler mit sagen wir nur drei Widerständen einmal durchrechnen, so ergibt sich ein ziemlich großer Fehler. An einem Spannungsteiler aus einem 10K und einem 1K Widerstand fällt eben am 1K Widerstand bei 10 Volt Eingangsspannung nicht 1 Volt sondern nur 0.909 Volt ab. Rechnen Sie es mal durch. Richtige Werte wären 9K und 1K oder 10K und 1.11111K.

Mit dem skizzierten Spannungsteiler läßt sich schon ein recht universell einsetzbares Voltmeter realisieren. Wenn aber in einer bestimmten Anwendung nur Spannungen bis maximal 30 Volt gemessen werden sollen, so könnte mit dieser Beschaltung nur bis auf ein Volt genau gemessen werden. Da wir aber über einen C64 die Werte anzeigen, ist es sehr einfach, den Messbereich auf diesen Bereich festzulegen und mit der maximalen Genauigkeit zu messen. Dazu dimensionieren wir den Spannungsteiler um. Wenn ein Messbereich genügt, so sieht der Spannungsteiler wie folgt aus.

Bei diesen Werten bleibt am 10K Widerstand ein Spannungsabfall von 1 Volt bei 30 Volt Eingangsspannung. Der vom Voltmeter gelieferte Wert muß jetzt natürlich mit 30 multipliziert werden. Diese Aufgabe kann der C64 aber im Programm leicht übernehmen.



Soviel zur Spannungsmessung. An Hand der Beispiele können Sie jetzt eigene Spannungsteiler nach Bedarf zusammenstellen.

7.4.2 Die Messung von Gleichströmen

Bei der Messung von Gleichströmen müssen wir einen kleinen Umweg beschreiten. Für diese Aufgabe müssen wir den Strom in eine äquivalente Gleichspannung umsetzen. Dazu eignet sich ein ganz normaler Widerstand. Sobald ein Gleichstrom durch den Widerstand fliesst, wird über ihm ein Spannungsabfall erzeugt, den wir messen können. Wie groß muß nun so ein Widerstand sein?
Nun, die Berechnung ist sehr einfach. Da unser DVM einen kleinsten Messbereich von 1 Volt hat, muss für einen bestimmten größten Strom genau ein Volt über dem Messwiderstand abfallen. Soll z.B. ein Strom von max. 1 Ampere gemessen werden, dann muß der Widerstand nach dem Ohmschen Gesetz genau 1 Ohm betragen. Zur Strommessung muss der Widerstand einfach in die zu messende Leitung eingeschleift werden. Da der Spannungsabfall immer 1 Volt beträgt, ist die Größe des Widerstands der Kehrwert des Stroms, d.h. bei 10 Ampere 0.1 Ohm, bei 0.1 Ampere 10 Ohm.

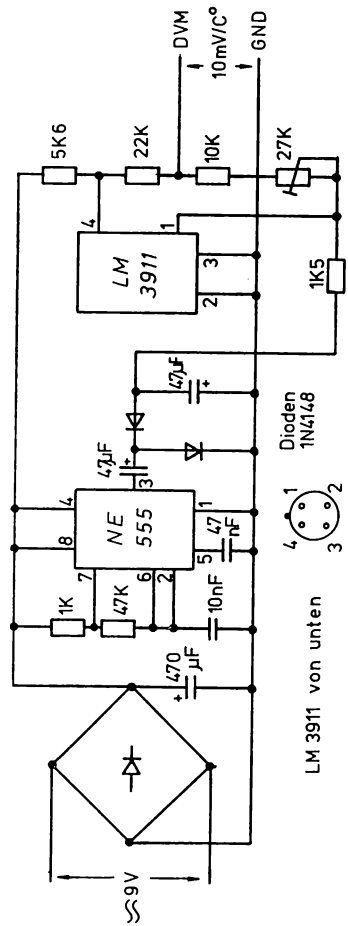
7.4.3 Die Temparaturmessung

Zur Temparaturmessung ist ein einfacher Widerstand als Wandler nicht geeignet. Für diesen Zweck hat die Bauteileindustrie verschiedene Ausführungen entwickelt. Die bekanntesten sind wohl der NTC und der PTC. Diese Bauteile stellen auch Widerstande dar, diese haben aber ein ausgeprägte Temparaturverhalten. So erhöht der PTC seinen Widerstandswert bei Erwärmung, beim NTC sinkt der Widerstand. Da diese Bauteile leider keine lineare Kennlinie besitzen, müßte für eine gute Linearität ein umständlicher Abgleich durchgeführt werden. Die modere Halbleiterindustrie liefert aber Temparatur-Spannungswandler, die mit nur einem Abgleich-Punkt auskommen und sehr genaue Werte liefern.
Für unseren Anwendungsfall gut geeignet ist das IC LM 3911. Mit diesem IC läßt sich recht einfach ein genaues Digital-Thermometer aufbauen.
Die vorgestellte Schaltung eignet sich in Verbindung mit dem CA 3162 ohne zusätzlichen Vorteiler für einen Meßbereich von -9.9 bis +99 Grad Celsius.
Das verwendetet IC ist in zwei Bauformen erhältlich, einmal im 8-poligen DIL-Gehäuse, einmal im 4-poligen Metall-Gehäuse. Die Version im Metall-Gehäuse hat den Vorteil, daß der eigentliche Temparatursensor gut in der Spitze eines ausrangierten Kugelschreibers untergebracht werden kann. Damit haben Sie ein handliches Gehäuse, mit dem Sie überall hin kommen und auch Oberflächentemparaturen an anderen Halbleitern messen können. Auch sind die Anschlüsse des ICs gut gegen Kurzschluß geschützt.
Das zusätzliche IC 555 wird benutzt, um eine negative Versorgungsspannung zu erzeugen. Dadurch können wir auch negative Temparaturen, also unter 0 Grad messen.
Der Abgleich ist denkbar einfach.

Dazu haben Sie zwei Alternativen. Die erste Alternative ist die Eiswasser-Methode. Dazu nehmen Sie eine Schüssel mit Eiswürfeln und lassen diese auftauen. Wenn sich nach etwa 5 bis 10 Minuten Schmelzwasser in der Schüssel sammelt, dann hat dies Wasser eine sehr genaue Temparatur von 0 Grad.

Wenn der Fühler in dies Wasser getaucht wird, (Anschlüsse des Fühlers gut isolieren) dann stellen Sie die Ausgangsspannung auf möglichst genau 0 Volt ein.

Die zweite Methode ist die Benutzung eines genauen Referenzthermometers. Ein solches Thermometer sollte in jedem Haushalt in Form eines Fieberthermometers vorhanden sein. Fieberthermometer sind bei der Herstellung auf 0.1 Grad geeicht. Damit können Sie die Temparatur in handwarmem Wasser bestimmen und die Ausgangsspannung der Schaltung einstellen.
Die Ausgangsspannung erhöht sich bei einer Temparaturerhöhung um 1 Grad sehr exakt um 10 Millivolt. Bei einer Temparatur von z.B. 37.5 Grad sollten Sie die Ausgangsspannung also auf 375 Millivolt einstellen.



8.0 Grundsätzliches zur Interrupttechnik

Das von vielen Computerbenutzern am meisten gemiedene Gebiet ist die Programmierung von Interrupts. Dabei spielt sicher eine Rolle, daß diese Programmierung in Maschinensprache erfolgen muß. Mit der Nutzung der Interrupts steht dem Programmierer aber ein sehr mächtiges Werkzeug zur Hand, mit dem Problemlösungen möglich sind, die ohne die Interrupts nicht oder nur schwer möglich wären.
Wenn Sie das vorliegende Buch bisher von Anfang an durchgearbeitet haben, dann haben Sie im Kapitel 4.4.2 bereits die beiden Interruptarten NMI und IRQ kennengelernt. Die dort gezeigte Stopuhr ließe sich ohne Interrupttechnik garnicht verwirklichen.

Was sind nun aber Interrupts ganz genau?
Nun, Interrupt bedeutet übersetzt Unterbrechung. Durch einen Interrupt wird das laufende Programm unterbrochen. Jetzt kann der Prozessor durch ein entsprechendes Programm überprüfen, was den Interrupt erzeugt hat. Anschließend wird auf den Interrupt reagiert, also die Quelle des Interrupts beseitigt und das zuvor unterbrochene Programm kann an der Stelle der Unterbrechung fortgesetzt werden.

Ein kleines Beispiel soll diesen Vorgang etwas genauer erklären. Dazu denken wir uns einmal einen C64, der eine Alarmanlage überwacht. Herkömmlich programmiert ist der Computer ausschliesslich mit der Überwachung beschäftigt und hat für andere Aufgaben keine Zeit.
Dieses Verfahren, das auch POLLING genannt wird, ist aber sehr unökonomisch. Wollten Sie zur selben Zeit den C64 für andere Aufgaben nutzen, so müßten Sie entscheiden, welche Aufgabe wichtiger ist. Als Lösung bietet sich gerade bei solchen Überwachungsproblemen die Interrupttechnik an.

Mit einer entsprechenden Hardware können Sie einen Interrupt erzeugen, wenn die Alarmanlage ausgelöst wird. Jetzt können Sie den Rechner genau so benutzen, als würde er die Überwachung überhaupt nicht vornehmen. Nur wenn ein Interrupt auftritt, wird die gerade bearbeitete Aufgabe unterbrochen, und der C64 kann einen akustischen oder/und einen optischen Alarm geben. Danach kann das unterbrochene Programm fortgesetzt werden.

Aber auch andere Arten der Unterbrechungstechnik sind möglich. So kann z.B. in bestimmten Zeitabständen ein Timer einen Interrupt erzeugen. In der Interruptroutine kann dann geprüft werden, ob ein bestimmtes Ereignis aufgetreten ist. Je nach Ergebnis kann dann im Programm unterschiedlich reagiert werden, um anschließend die Arbeit am laufenden Programm wieder aufzunehmen.
Diese Art der Interrupttechnik, der sogenannte Timerinterrupt, wird im C64 verwendet, um die Tastatur abzufragen. Alle 16 Millisekunden wird durch den Timer A der CIA Ul ein Interrupt ausgelöst. Dann wird überprüft, ob auf der Tastatur eine Taste gedrückt ist. Auch werden die Systemvariablen TI und TI$ über die Variablen in 'Schwung' gehalten.

Alle Leser, die gerne einen Beweis für die Interruptsteuerung in ihrem C64 haben wollen, können sich mit dem folgenden Programm davon überzeugen.

```
100 PRINT CHR$(147)
110 C1 = 56320
120 IR = C1 + 13
130 PRINT "TIPPEN SIE JETZT EINEN TEXT"
140 TI$ = "000000"
150 POKE IR,127
```

```
160 FOR I = 1 TO 10000
170 : GET A$ : Z$ = Z$ + A$
180 NEXT I
190 PRINT "JETZT NICHTS MEHR EINGEBEN"
200 FOR I = 1 TO 1000
210 NEXT I
220 POKE IR,129
230 PRINT Z$
240 PRINT TI$
```

Wenn Sie dies Programm einmal laufen lassen, werden Sie sehen, daß keine Daten von der Tastatur angenommen wurden. Auch steht die Uhr TI$ immer noch auf 000000. Lassen Sie zum Vergleich das Programm doch einmal ohne die Zeilen 150 und 220 laufen. Der Unterschied ist enorm.

Damit haben wir auch gleich festgestellt, wie sich vom Basic aus die Interrupts sperren lassen. Allerdings haben wir nicht wie in Maschinensprache den Interrupt per Befehl (SEI) verboten, sondern die Interruptquelle beseitigt. Dazu wurde das Interruptcontroll-Register in der CIA gelöscht. In der Zeile 220 wird dagegen der Interrupt durch Timerunterläufe von Timer A wieder zugelassen.

Der 6510 kennt drei verschiedene Interrupts, auch Interruptebenen genannt. Diese drei Ebenen haben unterschiedliche Priorität.
Die höchste Priorität besitzt der Reset. Ja, auch der Reset ist eine Art Interrupt. Sobald der Pin 40 des 6510 nach Masse gezogen wird, wird mit der Bearbeitung des gerade laufenden Programms aufgehört. Allerdings ist eine Rückkehr in das unterbrochene Programm nicht möglich.
Diese Rückkehr klappt aber beim nicht maskierbaren Interrupt NMI. Dieser durch Software nicht zu verbietende Interrupt

hat die zweithöchste Priorität. Er wird sofort ausgeführt, wenn der Pin 4 des 6510 nach Masse gelegt wird. Im C64 kann ein NMI auf zwei verschiedene Arten ausgelöst werden. Da ist zum einen eine Methode, die von jedem Besitzer eines C64 oft benutzt wird. Gemeint ist der Druck auf die RESTORE-Taste. Wichtig ist dabei, daß der NMI nur durch den Druck auf die RESTORE-Taste ausgelöst wird. Erst die durch den NMI angesprungene Routine prüft, ob gleichzeitig die RUN/STOP-Taste gedrückt ist.
Die zweite Methode besteht in einer entsprechenden Programmierung der CIA U2. Hier können alle in der Beschreibung der CIA genannten Möglichkeiten zur Interrupterzeugung genutzt werden.
Die dritte Interruptebene ist der sogenannte IRQ, Interrupt ReQuest. Dieser Interrupt kann durch ein Low-Signal an Pin 3 des 6510 ausgelöst werden.
Eine Besonderheit dieser Unterbrechung ist die Tatsache, daß durch Ausführung eines Maschinensprache-Befehls (SEI, Set Interruptflag) Low-Signale an diesem Pin des Prozessors ignoriert werden können. Man sagt, der Interrupt sei maskierbar.
Durch den Befehl CLI (Clear Interruptflag) kann diese Verriegelung wieder gelöst werden. Damit kann also im jeweiligen Programm entschieden weden, ob zu einem bestimmten Zeitpunkt IRQs zugelassen sind.
Im C64 kann der IRQ durch die CIA Ul ausgelöst werden. Wie schon erwähnt läuft hierüber der Timerinterrupt. Aber auch der Video-Controller 6569 kann bei bestimmten Ereignissen einen IRQ auslösen. So könnte beim Strahldurchlauf durch eine bestimmte programmierbare Zeile oder bei Sprite-Kollisionen ein IRQ ausgelöst werden.

Der Vollständigkeit halber sei an dieser Stelle noch eine softwaremäßige Möglichkeit zur Interrupterzeugung erwähnt.

Es ist dies der Maschinensprache-Befehl BRK. Wird in einem Programm dieser Befehl ausgeführt, so verhält sich der Prozessor etwa wie bei einem IRQ. Da dieser Befehl normalerweise nur beim Austesten von Maschinenprogrammen sinnvoll eingesetzt wird, wollen wir auf eine detailierte Beschreibung verzichten.

Zusätzlich zu den bisher genannten Möglichkeiten der Unterbrechungserzeugung im C64 haben wir die Möglichkeit, einen Interrupt durch eine externe Baugruppe auszulösen, da sowohl der NMI als auch der IRQ auf den Expansion-Port herausgeführt sind.
Doch sehen wir uns nun einmal an, wie wir die Interrupts für unsere Aufgaben nutzen können.

8.1 Es laufen zwei Dinge gleichzeitig

Nachdem wir im vorigen Kapitel gesehen haben, was es mit der Interrupttechnik auf sich hat und welche Möglichkeiten der C64 bietet, wollen wir nun einige kleine Programme zeigen, die Ihnen das Verständnis hoffentlich etwas erleichtern.
Bevor wir aber mit der Programmierung beginnen, müssen noch einige Dinge erklärt werden.
Für die Abarbeitung der Unterbrechungen gibt es im Betriebssystem des C64 einige Speicherzellen, die als Zeiger oder Vektoren auf die eigentlichen Interruptroutinen zeigen. Diese Vektoren liegen im RAM. Dabei belegt der IRQ-Vektor die Adressen $0314 und $0315, der NMI-Vektor belegt die Adressen $0318 und $0319. Je nach auftretendem Interrupt wird von diesen Adressen die Adresse der eigentlichen Interruptroutine geholt.
Da diese Vektoren im RAM liegen, haben wir die Möglichkeit, hier die Adresse einer eigenen Interruptroutine zu hinterlegen. Tritt nun eine Unterbrechung auf, so wird automatisch in unser Programm verzweigt. Jetzt haben wir die Möglichkeit zu prüfen, ob der aufgetretene NMI oder IRQ für uns gemeint ist, oder ob es sich dabei um den Systeminterrupt handelt. Stellen wir fest, daß wir nicht gemeint sind, verzweigen wir einfach in die alte Interruptroutine, an sonsten führen wir die benötigten Aktivitäten durch.
Das hört sich alles schrecklich theoretisch an. Da man in der Praxis am besten etwas lernt, sollten Sie das nun folgende Programm gründlich studieren. In diesem Programm wollen wir vom sogenannten Rasterinterrupt Gebrauch machen. Dabei wird der Bildschirm in drei Abschnitte geteilt.Die oberen und unteren 6 Zeilen werden mit blauer Rahmen- und Hintergrundfarbe dargestellt, die mittleren 13 Zeilen erscheinen als weißer Rahmen und Hintergrund. Das ist keine

sehr aufregende Aufgabe, wenn sie aber das Prinzip verstanden haben, ist ein wichtiger Schritt zur Beherschung der Interrupts getan.

```
0001          WEISS    =    1          ;hier können Sie eigene
0006          BLAU     =    6          ;Farben wählen
EA31          IRQALT   =    $EA31      ;Systeminterrupt
0314          IRQVECT  =    $0314      ;Vektor im Ram
D012          RASTER   =    $D012      ;Rasterzeilenreg. des 6569
D019          IRQREG   =    $D019      ;Interruptregister
D01A          MASKE    =    $D01A      ;Interruptmaske
D020          RAHMEN   =    $D020      ;bestimmt Rahmenfarbe
D021          HINTERG  =    $D021      ;bestimmt Hintergrund
0062          RZ1   =    98            ;Rasterzeilen, bei denen
00C9          RZ2   =    201           ;Interrupt ausgelöst wird
                                       ;Start des Programms
C000 78       SETUP    SEI             ;weitere Interrupts sperren
C001 A9 1F             LDA  #< IRQNEU  ;Interruptvektor auf unsere
C003 8D 14 03          STA  IRQVECT    ;Routine verbiegen
C006 A9 C0             LDA  #> IRQNEU
C008 8D 15 03          STA  IRQVECT+1
C00B A9 62             LDA  #RZ1       ;erste Rasterzeile
C00D 8D 12 D0          STA  RASTER     ;initialisieren
C010 AD 11 D0          LDA  RASTER-1   ;siehe Text
C013 29 7F             AND  #$7F
C015 8D 11 D0          STA  RASTER-1
C018 A9 81             LDA  #$81       ;Rasterinterrupts
C01A 8D 1A D0          STA  MASKE      ;zulassen,
C01D 58                CLI             ;Interrupts freigeben
C01E 60                RTS             ;und zurück
                                       ;hier beginnt die neue
                                       ;Interruptroutine
C01F AD 19 D0 IRQNEU   LDA  IRQREG     ;kommt der IRQ vom
C022 8D 19 D0          STA  IRQREG     ;6569 ?
```

```
C025 29 01            AND  #1
C027 D0 07            BNE  DOIT       ;wenn ja, dann DOIT
C029 AD 0D DC         LDA  $DC0D      ;CIA-IRQ löschen
C02C 58               CLI             ;und zur Systemroutine
C02D 4C 31 EA         JMP  IRQALT     ;verzweigen
C030 AD 12 D0 DOIT    LDA  RASTER     ;IRQ durch zweite
C033 C9 C9            CMP  #RZ2       ;Rasterzeile ?
C035 B0 10            BCS  ZWEITER    ;dann nach ZWEITER
C037 A9 01            LDA  #WEISS     ;sonst Rahmen und Hinter-
C039 8D 20 D0         STA  RAHMEN     ;grund auf Weiß schalten
C03C 8D 21 D0         STA  HINTERG
C03F A9 C9            LDA  #RZ2       ;IRQ durch zweite RZ
C041 8D 12 D0         STA  RASTER     ;vorbereiten und
C044 4C BC FE         JMP  $FEBC      ;IRQ-Routine verlassen
C047 A9 06    ZWEITER LDA  #BLAU      ;Rahmen und Hintergrund
C049 8D 20 D0         STA  RAHMEN     ;auf blau schalten und
C04C 8D 21 D0         STA  HINTERG
C04F A9 62            LDA  #RZ1       ;auf 1.RZ vorbereiten
C051 8D 12 D0         STA  RASTER
C054 4C BC FE         JMP  $FEBC      ;IRQ verlassen
```

Nach dem Laden wird das Programm mit SYS 12 * 4096 gestartet.
Für die Leser ohne Assembler folgt der Basic-Loader. Die unterstrichenen Werte in den Datazeilen sind für die Farbe und die Rasterzeilen maßgeblich. Wenn Sie hier andere Werte einsetzen,dann bedenken Sie bitte, das Sie Farbwerte von 0 bis 15 und die Rasterzeilen von 0 bis 255 wählen können. Außerdem verändert sich bei anderen Rasterzeilen- und Farbwerten die Prüfsumme. Aus diesem Grund sollten Sie das Programm erst einmal in der originalen Fassung eingeben und danach die Werte ändern. Wenn dann nach dem RUN ein 'Fehler in den Datas' auftritt, können Sie mit 'PRINT S' die neue Prüfsumme erfahren.

Die Rasterzeilen haben zwar ein Übertragsbit im Register $D011, dieses Bit wird in unserem Programm aber nicht verwendet. Es wird schon bei der Initialisierung gelöscht.

```
100 FOR I = 49152 TO 49238
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 120,169, 31,141, 20,  3,169,192,141, 21,  3,169
130 DATA  98,141, 18,208,173, 17,208, 41,127,141, 17,208
140 DATA 169,129,141, 26,208, 88, 96,173, 25,208,141, 25
150 DATA 208, 41,  1,208,  7,173, 13,220, 88, 76, 49,234
160 DATA 173, 18,208,201,201,176, 16,169,  1,141, 32,208
170 DATA 141, 33,208,169,201,141, 18,208, 76,188,254,169
180 DATA   6,141, 32,208,141, 33,208,169, 98,141, 18,208
190 DATA  76,188,254
200 IF S <> 10596 THEN PRINT "FEHLER IN DATAS !!" : END
210 PRINT "OK"
220 SYS 49152
```

An Hand dieses Beispiels können wir zwar sehr schön sehen, wie Interrupts programmiert werden, für Meß- und Überwachungsaufgaben ist die Sache etwas anders. Wenn Sie mit einem A/D-Wandler eine Spannung überprüfen, dann kann der A/D-Wandler ja keinen Interrupt auslösen, da er für einen Interruptbetrieb nicht vorgesehen ist. In diesem Fall muß man einen anderen Weg beschreiten.
Durch Ausnutzung des Timerinterrupts können wir alle 16 Millisekunden nachsehen, ob die zu überwachende Spannung im Sollbereich liegt. Solange das der Fall ist, können wir sofort in die alte Systemroutine zur Unterbrechungshandhabung verzweigen. An sonsten kann eine nach der jeweiligen Anwendung erforderliche Aktion eingeleitet werden. Wir haben einmal eine kleine Routine geschrieben, die bei jedem Interrupt den User-Port liest. Liegt der eingelesene Wert im Bereich zwischen 0 und 150, so erscheint

der Rahmen in Grün. Bei Werten zwischen 150 und 200 wird der Rahmen hell rot, bei darüberliegenden Werten dunkelrot. Interessant wird dies Programm mit einem angeschlossenen A/D-Wandler nach Kapitel 7.3.

```
0005            GRUEN    =   5
000A            HELLROT  =   10
0002            ROT      =   2
EA31            IRQALT   =   $EA31
0314            IRQVECT  =   $0314
DD00            CIA2     =   $DD00
DD01            PB       =   CIA2 + 1
DD03            DR       =   CIA2 + 3
D020            RAHMEN   =   $D020
                                      ;hier geht's los
C000 78         SETUP    SEI
C001 A9 12               LDA  #< IRQNEU ;Interruptvektor auf
C003 8D 14 03            STA  IRQVECT   ;unsere Routine
C006 A9 C0               LDA  #> IRQNEU ;verbiegen
C008 8D 15 03            STA  IRQVECT+1
C00B A9 FF               LDA  #$FF      ;User-Port auf Eingang
C00D 8D 03 DD            STA  DR
C010 58                  CLI            ;Interrupts freigeben
C011 60                  RTS            ;und zurück
                                        ;hier beginnt die
                                        ;eigentliche IRQ-Routine
C012 AD 01 DD IRQNEU     LDA  PB        ;User-Port lesen
C015 C9 C8               CMP  #200      ;über 200?
C017 90 04               BCC  W1        ;nein dann nach W1
C019 A2 02               LDX  #ROT      ;ja dann X mit 2 (rot)
                                        ;laden
C01B D0 0A               BNE  WW        ;und nach WW springen
C01D C9 96      W1       CMP  #150      ;über 150?
C01F 90 04               BCC  W2        ;nein dann nach W2
```

```
C021 A2 0A          LDX  #HELLROT  ;ja dann hellrot
C023 D0 02          BNE  WW
C025 A2 05    W2    LDX  #GRUEN    ;Wert ist <150, also gruen
C027 8E 20 D0 WW    STX  RAHMEN    ;in das Rahmenregister
C02A 4C 31 EA       JMP  IRQALT    ;und zur alten IRQ-Routine
```

Jetzt kann man wirklich sagen, das zwei Programme 'gleichzeitig' laufen. Unabhängig von anderen Programmen wird nach dem Aufruf des Programms mit SYS 12 * 4096 der User-Port überwacht. Sie können jetzt ohne weiteres andere Programme laden und ausführen. Allerdings dürfen diese Programme den Interruptvektor nicht verändern, die Ergebnisse wären nicht vorhersehbar.
Zum Abschluß noch den zum Programm gehörenden Basic-Loader.

```
100 FOR I = 49152 TO 49196
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 120,169, 18,141, 20,  3,169,192,141, 21,  3,169
130 DATA 255,141,  3,221, 88, 96,173,  1,221,201,200,144
140 DATA   4,162,  2,208, 10,201,150,144,  4,162, 10,208
150 DATA   2,162,  5,142, 32,208, 76, 49,234
160 IF S <> 5085 THEN PRINT "FEHLER IN DATAS !!" : END
170 PRINT "OK"
180 SYS 12 * 4096
```

Betrachten Sie die abgedruckten Programme bitte nur als Aufforderung zum Experimentieren. Es sollten nur die grundsätzlichen Möglichkeiten aufgezeigt werden. Wenn Sie die Interrupts verstanden haben, werden Sie feststellen, daß damit wirklich interessante Problemlösungen möglich sind.

9.0 Komplette Schaltungen zum Selberbauen

In diesem letzten Kapitel werden wir einige besonders interessante und in dieser Form unseres Wissens nach für den C64 noch nicht publizierte Schaltungen besprechen.

Dabei werden wir auch ganz gezielt die verschiedenen speziellen hardwarebedingten Eigenheiten des C64 nutzen.

Zu allen Schaltungen haben wir soweit nötig eine etwas umfangreichere Software geschrieben, die das Arbeiten mit den Zusätzen möglichst komfortabel gestaltet. Soweit möglich wurde aus Gründen der Transparenz der Funktion des Programms in Basic programmiert. Nur in zeitkritischen Programmteilen wurde in Maschinensprache programmiert.

Die Hardwarekosten wurden bewußt niedrig gehalten, um auch jüngeren Lesern mit schmalen Geldbeutel den Nachbau zu ermöglichen.

9.1 Eprom-Programmiergerät

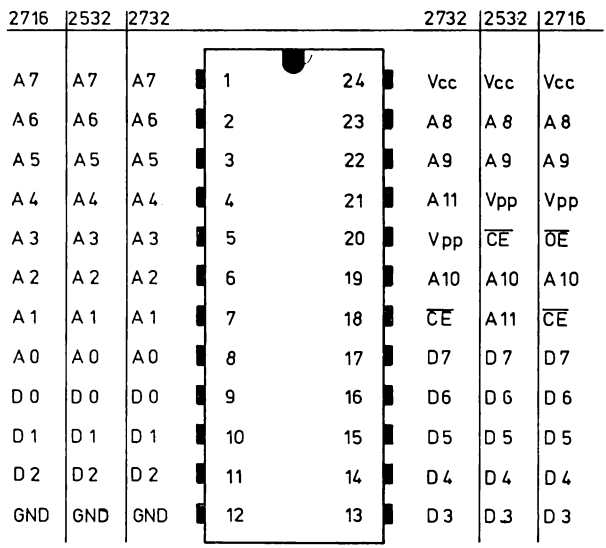
Wozu überhaupt ein Programmiergerät für den C64 werden Sie jetzt vielleicht fragen.
Nun, mit fortschreitender Erfahrung auf dem Gebiet der Microcomputertechnik (vielleicht auch mit Hilfe dieses Buchs) werden Ihre Anwendungen für den Computer immer komplexer werden. Über kurz oder lang kann dann der Zeitpunkt kommen, daß Sie die ganzen Aufgaben nicht mehr nur mit Ihrem C64 lösen können oder möchten. Dann empfielt sich der Einsatz von sogenannten Einplatinencomputern.
Diese Kleinstcomputer eignen sich ganz ausgezeichnet zum Aufbau von Steuerungen. Die benötigte Software muß bei diesen Rechnern in Form von Eproms vorliegen.
Mit einem Assembler, einem Monitorprogramm und dem Prommer wird nun aus Ihrem C64 ein ganz passables Entwicklungssystem für die Software dieser Steuerungscomputer.
Aber auch andere Möglichkeiten sind mit dem Prommer gegeben. So könnten Sie einen Maschinensprache-Monitor in ein Eprom brennen. Dann erübrigen sich langwierige Ladezeiten. Sie stecken den Monitor wie ein Spielemodul in den Expansion-Slot und schon steht der Monitor zur Vefügung.
Als letzte Möglichkeit sei erwähnt, daß in dem Eprom diverse Testroutinen untergebracht werden können. So können ein Ramtest, ein Test der Peripherie-ICs und ein Test der verschiedenen Schnittstellen programmiert werden. Sollte dann der C64 irgendwann einmal einen Defekt zeigen, so stekken Sie nur das Eprom-Modul in den Expansion-Slot. Solange der Rechner überhaupt noch arbeitet, kann er jetzt die Testroutinen ausführen und eventuelle Fehler auf dem Bildschirm anzeigen. Dabei können Sie natürlich von der Möglichkeit des Auto-Starts Gebrauch machen. Auf diese Möglichkeit werden wir später noch detailliert zu Sprechen kommen.
Der vorgestellte Prommer ermöglicht die Programmierung der

drei wichtigsten Eprom-Typen. Programmiert werden 2716, 2532 und 2732.

Bei der Entwicklung der Schaltung wurde auf eine besonders einfache Handhabung Wert gelegt. So enthält die Schaltung keinen Schalter. Alle Funktionen werden Softwaremäßig durchgeführt. Die Wahl der zu programmierenden Epromtype erfolgt durch ein kleines Steckmodul. Dieses Modul kann einfach aus einem 14-poligen IC-Sockel hergestellt werden.

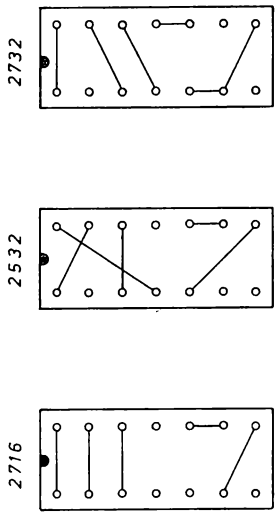
Das Steuerprogramm kann durch einen kleinen Schaltungstrick sogar feststellen, welches Modul eingesteckt ist. Dadurch wird automatisch die richtige Eprom-Type ausgewählt.

Das untenstehende Bild zeigt die Unterschiede in der Anschlußbelegung der drei Epromtypen. Durch eine ausgeklügelte Schaltung konnten alle Umschaltungen der Anschluß-Beinchen über das Modul vorgenommen werden.



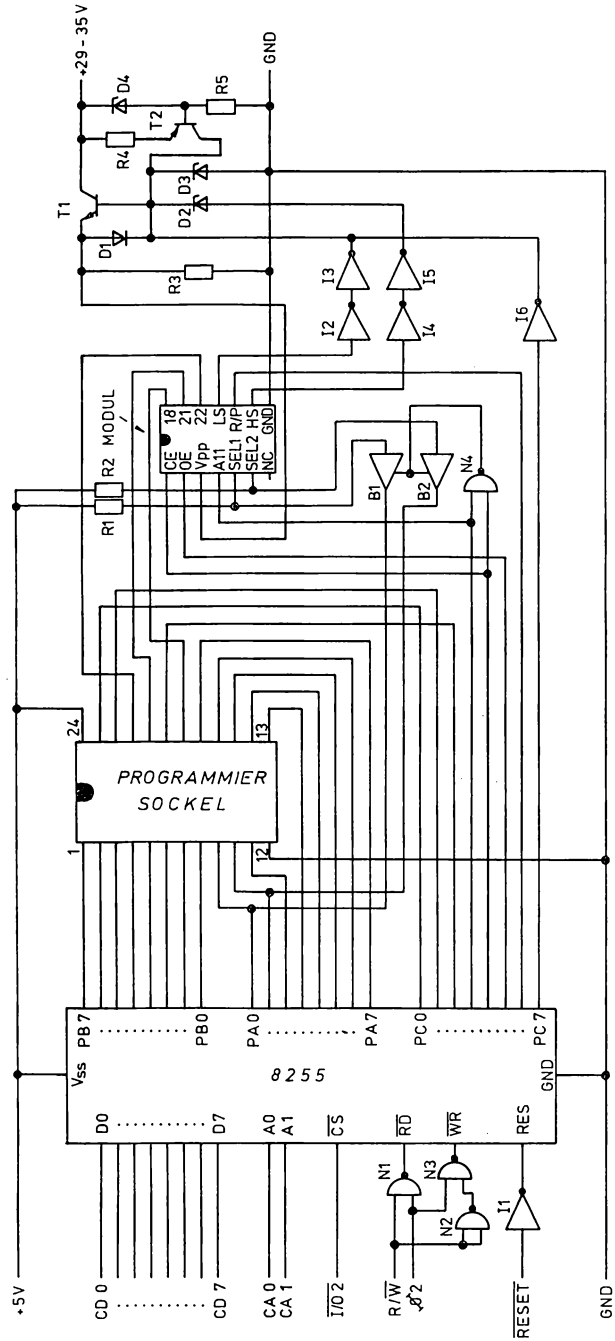
Die Hardware des Prommers ist rund um ein 8255 aufgebaut. Wie wir schon bei der Beschreibung des 8255 gesehen haben, bietet dies IC drei vollständige 8-Bit-Ports. Das sind genau 24 I/O-Leitungen. Da die zu programmierenden Eproms auch 24 Anschlüße haben, können wir den 8255 vollständig ausnutzen.
Der Port A des 8255 wird von uns als Datenbus verwendet. Über diese Leitungen werden die Daten eines schon programmierten Eproms in den Rechner gelesen, beim Programmieren dagegen werden hierüber die Daten dem Eprom zugeführt.
Zusätzlich wird über die Bits 0 und 1 eingelesen, um welchen Epromtyp es sich handelt. Diese Information wird vom Modul geliefert.
Der Port B stellt die unteren 8 Adressbits des zu programmierenden Eproms dar. Da diese Adressbits bei den verwendeten Eproms auf den gleichen Anschlüssen liegen, gibt es hierbei keine Besonderheit.
Port C übernimmt verschiedene Aufgaben.
Zum einen werden über die Bits 0 - 2 die Adressbits A8 bis A10 erzeugt.
PC3 erzeugt das Adressbit A11. Dieses Adressbit wird nur beim 2532 und beim 2732 benötigt. Da dies Adressbit bei den beiden Typen aber auf unterschiedlichen Anschlüßen liegt, muß das Modul die entsprechende Verbindung herstellen.
PC4 bildet das Signal -CE, Chip Enable. Zusätzlich zur Funktion -CE wird beim Programmieren der Programmierimpuls von 50 Millisekunden von diesem Portbit erzeugt. Auch dies Signal wird über das Modul verschaltet, je nach verwendetem Eprom.
-OE, das von PC5 erzeugte Signal kommt nur bei dem 2k-Eprom 2716 zum Einsatz. Auch dies Signal liegt am Modulstecker zur Auswahl an.
PC6 schaltet mit einem High den Prommer in den Programmierzustand, bei einem Low können die Daten aus dem

Verdrahtung der Module



Bauteileliste

R1	3K3
R2	3K3
R3	10K
R4	1K
R5	3K3
D1	1N4148
D2	ZPD5,6
D3	ZPD27
D4	ZPD5,6
N1-4	7400
B1,2	74125
I1-6	7416
T1	BC 238
T2	BC307



Eprom gelesen werden.
PC7 zu guter Letzt schaltet die Programmierspannung ein und aus. Sie beträgt 25 Volt. Sollen die in letzter Zeit vermehrt erhältlichen 21 V-Eproms geschossen werden, so tauschen Sie die Z-Diode D3 (27 V) gegen eine solche mit einer Zenerspannung von 22 Volt. Durch die Basis-Emitter-Spannung des Transistors T1 reduziert sich die Spannung auf den gewünschten Wert von etwas mehr als 21 Volt.
Damit sind wir auch schon bei der Beschreibung der Hardware. Außer dem 8255 sind vorhanden ein 74LS125, ein 7400, ein 7416 und einige Dioden, Transistoren und Widerstände.
Die Transistoren T1 und T2 bilden eine Stabilisierungsschaltung für die Programmierspannung. Diese Stufe kann am Ausgang eine Spannung von 0 Volt, 5 Volt oder 25 Volt liefern. Die 5 Volt werden für die Eproms 2716 und 2532 benötigt. Bei diesen Eproms muß der Anschluß für die Programmierspannung beim Lesen der Daten auf 5 Volt gehalten werden. Das Eprom 2732 wünscht für ordnungsgemäßen Betrieb beim Lesen aber 0 Volt am Programmiereingang. Diese Spannungsumschaltung wird mit 5 Invertern realisiert.
Wird am Modul der Pin HS nach Masse gelegt und ist PC7 Low, dann liegt die Z-Diode D2 über den Inverter I5 an Masse. Diese Z-Diode mit einem Wert von 5.7 Volt erzeugt nun eine Ausgangsspannung von 5 Volt.
Ist dagegen LS Low und PC7 Low, dann wird die Z-Diode D3 durch den Inverter I3 kurzgeschlossen.
Wird PC7 Low und LS oder HS, also PC6, auf High programmiert, dann stellt sich am Ausgang die Programmierspannung von 25 Volt ein.
Durch ein High an PC7 wird unabhängig von PC6 der Ausgang auf 0 Volt geschaltet, da der Inverter I6 die Basis des Transistors T1 nach Masse zieht und den Transistor somit sperrt.
Um dieses Ein-, Aus- und Kurzschließen von Spannungen für

die Inverterausgänge gefahrlos zu ermöglichen, ist mit dem Transistor T2 eine Konstantstromquelle realisiert. Der maximale Kollektorstrom von T2 ist durch die Z-Diode D4 und den Widerstand R5 auf einen Wert von 5 Milliampere begrenzt. Die Stufe mit dem Nand-Gatter N4 und den beiden Tri-State-Buffern 74LS125 dient zur Bestimmung der Epromtype. Wenn PC3 und PC4 High sind, werden die Daten von den Eingängen der Buffer auf den Ausgang geschaltet. Wenn man jetzt einmal die Diagramme für die Programmierung der Eproms zur Hand nimmt, kann man sehen, daß dieser Zustand bei allen in Frage kommenden Eproms eine Deselektierung des Eproms zur Folge hat. Damit können keine Daten verfälscht werden.

Doch kommen wir nun zum Programm. Nach RUN wird zunächst eine kleine Maschinenroutine in den Speicher gePOKEd. Diese Routine benutzt einen Timer in der CIA, um den Programmierimpuls von 50 Millisekunden zu erzeugen. Die Zeitschleife läßt sich in Basic nicht realisieren.

Danach wird festgestellt, welches Programmiermodul eingesteckt ist. Sollte kein Modul eingesteckt sein, wird ein Fehler gemeldet und das Programm beendet. Ist dagegen alles in Ordnung, so erhalten Sie ein Menu auf dem Bildschirm. Sie können jetzt verschiedene Programmpunkte auswählen.

So können Sie prüfen, ob ein Eprom gelöscht ist. Diese Überprüfung empfiehlt sich vor jedem Programmieren.

Dann können Sie ein schon programmiertes Eprom in den Speicher einlesen, um es zu duplizieren, oder seinen Inhalt zu modifizieren.

Natürlich gibt es die Funktion Eprom programmieren. Nach dem Programmieren wird automatisch geprüft, ob der Eprom-Inhalt mit dem entsprechenden Speicherinhalt identisch ist. Sollte durch irgendeinen Umstand das Eprom nicht ordnungsgemäß programmiert worden sein, so wird ein Fehler angezeigt.

Die Funktion Vergleich Eprom - Speicherbereich ist auch als separater Menu-Punkt aufgeführt.
Um einen anderen Epromtyp als den eingestellten zu programmieren, ist der Menu-Punkt 6 vorhanden. Wechseln Sie also das Modul, so drücken Sie anschließend 6 oder Starten das Programm mit RUN neu.
Der letzte vorhandene Menu-Punkt gestattet den Ansprung eines Monitorprogramms. Damit haben Sie die Möglichkeit, den Speicherinhalt byteweise zu verändern. Bevor Sie diesen Punkt benutzen, müssen Sie natürlich zunächst den Monitor in den Speicher geladen haben.
Ein kurzes Wort zur Stromversorgung. Die Betriebsspannung von 5 Volt für die verwendeten ICs werden vom C64 geliefert. Die Programmierspannung muß jedoch von einer externen Stromversorgung zugeführt werden. Da die Spannung ja durch die Schaltstufe stabilisiert wird, kann zwischen 28 Volt und ca. 40 Volt jede Gleichspannung verwendet werden. Gut geeignet ist ein kleiner Trafo für direkte Platinenmontage mit einer Sekundärspannung von 24 Volt. Nach Gleichrichtung mit einem Brückengleichrichter und anschließender Siebung mit einem Elko von 470 uF erhalten wir eine Gleichspannung von etwa 33 Volt, also genau richtig. Die Stromaufnahme liegt bei Programmierung unter 50 Milliampere. Es genügt also der kleinste lieferbare Trafo.

```
5 REM         PROMMER V1.2
10 REM ******************************
15 REM **                            **
20 REM **     PROMMER SOFTWARE       **
25 REM **                            **
30 REM **  VERSION VOM 04.12.1983    **
35 REM **                            **
40 REM **      FUER   C  64          **
45 REM **                            **
50 REM ******************************
55 POKE56,128 : REM BASIC-ENDE HERABSETZEN
60 PP=57088   : REM PROMMERPORT
65 HR=52992   : REM $CF00 START 50 MILLI
70 RA=32760   : REM START DES RAMS FUER EPROM INHALT
80 FORI=1TO40:ST$=ST$+CHR$(162):NEXT
90 CU$=CHR$(145):CD$=CHR$(17):CR$=CHR$(29):CL$=CHR$(157)
95 FORI=1TO24:F1$=F1$+CR$:NEXT
96 F1$=CU$+F1$+"      "+CL$+CL$+CL$+CL$+CL$
100 DIM PM(3),EP(3),BZ(3),MA(3)
110 PM(1)=112  :PM(2)=64   :PM(3)=64
120 EP(1)=2716 :EP(2)=2532 :EP(3)=2732
130 BZ(1)=2048 :BZ(2)=4096 :BZ(3)=4096
140 MA(1)=96   :MA(2)=80   :MA(3)=80
150 TX$(1)="EPROM LESEN":
151 TX$(2)="EPROM PROGRAMMIEREN"
152 TX$(3)="EPROM = SPEICHER ?"
153 TX$(4)="EPROM GELOESCHT ?"
154 TX$(5)="SPRUNG ZU PROFIMON"
155 TX$(6)="NEUES EPROMMODUL EINLESEN"
160 FORI=0TO41
170 READ A
180 POKEHR+I,A
190 NEXT
```

```
250 POKEPP+3,144:REM ALLES AUSGANG
300 POKEPP+2,184:REM %11110000
350 EP=PEEK(PP) : REM MODUL LESEN
400 EP=-EP+255  : REM UND TYPE ERMITTELN
500 IFEP<1OREP>3THEN49000 : REM KEIN MODUL ?
550 PRINTCHR$(147):PRINT"          EPROMTYPE IST ";EP(EP)
600 PRINT:PRINT ST$
700 FORI=1TO6
750 PRINT"   "I". "TX$(I)
800 NEXT
1000 A=0
1050 PRINT:PRINT:PRINT"         BITTE FUNKTION WAEHLEN"
1060 PRINT:INPUT"                 ";A:IFA<1ORA>6THEN550
1500 ON AGOSUB2000,3000,4000,5000,6000,7000
1950 GOTO250
2000 POKEPP+3,144
2050 POKEPP+1,0
2150 POKEPP+2,0
2350 GOSUB20000
2400 ZA=AN
2500 FORBT=ESTOES+AN
2520 PRINTF1$;ZA
2550 HA=INT(BT/256):LA=BT-HA*256
2600 POKEPP+1,LA
2650 POKEPP+2,HA
2700 W=PEEK(PP)
2710 POKERS+BT,W
2800 ZA=ZA-1:NEXTBT
2999 RETURN
3000 POKEPP+3,128
3050 POKEPP+1,0
3150 POKEPP+2,0
3200 POKE53001,PM(EP)
3350 GOSUB20000
```

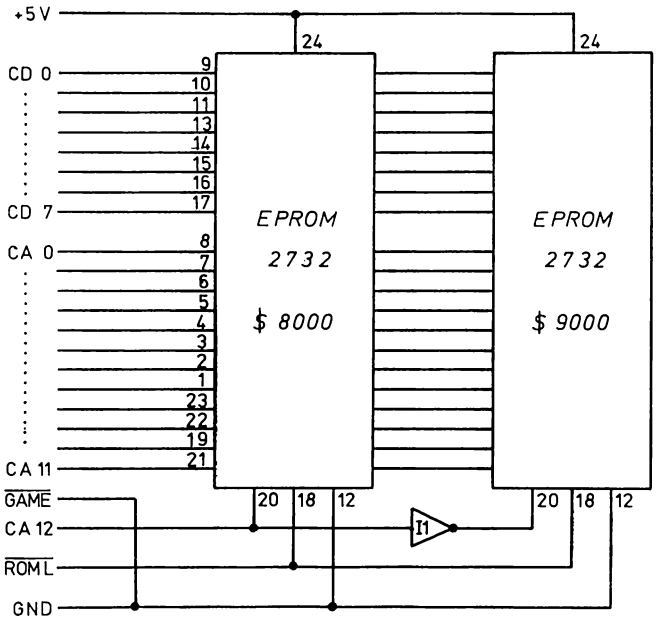
```
3400 ZA=AN
3500 FORBT=ESTOES+AN:PRINTFl$;ZA
3550 HA=INT(BT/256):LA=BT-HA*256
3560 HA=HAORMA(EP)
3600 POKEPP+l,LA
3650 POKEPP+2,HA
3700 W=PEEK(RS+BT)
3710 POKEPP,W
3750 SYS52992   : REM PROGRAMMIERIMPULS
3800 ZA=ZA-l:NEXTBT
4000 POKEPP+3,144
4050 POKEPP+l,0
4150 POKEPP+2,0
4300 IFA=2THENPRINTCD$;CD$;"   ";TX$(3);CU$;CU$;CU$:GOTO4400
4350 GOSUB20000
4400 ZA=AN
4500 FORBT=ESTOES+AN:PRINTFl$;ZA
4550 HA=INT(BT/256):LA=BT-HA*256
4600 POKEPP+l,LA
4650 POKEPP+2,HA
4700 W=PEEK(PP):
4710 IFW<>PEEK(RS+BT)THENBT=BZ(EP):FL=l
4750 IFFL=lTHENPRINTCD$;CD$;"   EPROM NICHT GLEICH "
4800 ZA=ZA-l:  NEXTBT
4850 IF FL=l THEN FL=0:GOTO4950
4900 PRINTCD$;CD$;"   EPROM IST GUT !!! "
4950 GOSUB50000
4999 RETURN
5000 POKEPP+3,144
5050 POKEPP+l,0
5150 POKEPP+2,0
5350 GOSUB20000
5400 ZA=AN
5500 FORBT=ESTOES+AN:PRINTFl$;ZA
```

```
5550 HA=INT(BT/256):LA=BT-HA*256
5600 POKEPP+1,LA
5650 POKEPP+2,HA
5700 W=PEEK(PP)
5710 IFW<>255THENPRINTCD$;CD$;"   EPROM NICHT LEER !!!!":BT=BZ(EP)
5800 ZA=ZA-1:  NEXTBT
5850 IFW=255THEN PRINTCD$;CD$;"   EPROM IST LEER"
5950 GOSUB50000
5999 RETURN
6000 SYS12*4096
7000 RUN
20000 PRINTCHR$(147):PRINT"          EPROMTYPE IST ";EP(EP)
20020 PRINT:PRINT"          ";:PRINTTX$(A):PRINT: PRINTST$:PRINT
20030 IFA =4THENGOTO20100
20050 PRINT:PRINT"   RAMADRESSE START      0";CL$;CL$;CL$;:INPUT RS:RS=RS+32768
20100 PRINT:PRINT"   EPROMADRESSE START    0";CL$;CL$;CL$;:INPUTES
20150 PRINT:PRINT"   ANZAHL DER BYTES     ";BZ(EP)-ES;:FORI=1TO7:PRINTCL$;:NEXT
20200 INPUTAN: AN=AN-1
20250 IFES>BZ(EP)THEN20000
20300 IFES<0THEN250
20350 IFAN>BZ(EP)THEN20000
20400 IFAN<0THEN250
20450 RETURN
49000 PRINTCHR$(147):PRINT:PRINT"          F E H L E R":PRINT:PRINTST$
49020 PRINT:PRINT:PRINT"  KEIN PROGRAMMIERMODUL ODER"
49040 PRINT:PRINT"  MODUL FALSCH EINGESTECKT !!!!"
49100 END
50000 PRINTCD$;CD$;"   RETURN FUER WEITER DRUECKEN"
50010 GETA$:IFA$<>CHR$(13)THEN50010
50020 RETURN
60000 REM  DATAS FUER MASCHINENSPRACHE
60010 DATA 120, 173,   2, 223
60020 DATA 133, 247,  41,  15
60030 DATA   9,   0, 141,   2
```

```
60040 DATA 223, 169,  80, 141
60050 DATA   4, 221, 169, 195
60060 DATA 141,   5, 221, 169
60070 DATA  25, 141,  14, 221
60080 DATA 169,   1,  44,  13
60090 DATA 221, 240, 251, 165
60100 DATA 247, 141,   2, 223
60110 DATA  88,  96
```

9.2 Eprom-Karte für den C64

So, nachdem Sie jetzt wohl Ihre ersten Eproms erfolgreich programmiert haben, wollen wir uns anschauen, wie ein Eprom an den C64 angeschlossen werden kann. Als Steckplatz kommt der Expansion-Slot in Frage. Hier liegen alle benötigten Bussignale.
Ein Blick in die Unterlagen zum C64 zeigt uns, daß der günstigste Adressbereich für den Betrieb von Eproms der Bereich zwischen $8000 und $9FFF ist.
Zum Ersten ist dieser Bereich mit 8K auch für umfangreiche Programme groß genug.
Außerdem ist dieser Bereich der sogenannte Autostart-Bereich. In diesem Bereich liegende Programme können nach dem Einschalten des Rechners direkt gestartet werden (wie, darüber später mehr).
Als letzter Vorteil ist zu sehen, daß in diesem Adressbereich keine speziellen Maßnahmen zur Adress-dekodierung getroffen werden müssen. Erreicht wird dies durch die Anschlüße -GAME und -ROML am Expansion-Slot. Sobald die Leitung -GAME nach Masse gezogen wird, kann das Signal -ROML als Chip Selekt Signal für ein Eprom verwendet werden. Das normalerweise in diesem Bereich liegende RAM ist dann ausgeblendet.
Da der in Frage kommende Bereich 8K groß ist, sind in der Schaltung Sockel für zwei Eproms vom Typ 2732 vorgesehen. Um ein Eprom im Adressbereich von $8000 bis $8FFF zu betreiben, sind keine zusätzlichen Gatter erforderlich. Die Adressierung geschieht über das Signal ROML für -CS (Chip Select) und zusätzlich das Adressbit 12 als -OE (Output Enable). Soll aber mit einem zweiten 2732 der Bereich von $9000 bis $9FFF abgedeckt werden, so müssen wir für das zweite Eprom das Adressbit 12 invertieren und als -OE-Signal verwenden.



Doch nun die Erklärung des Autostarts

Nach dem Einschalten durchläuft der C64 seine Einschalt- oder Resetroutine. In dieser Einschaltroutine wird als erstes der Interrupt gesperrt und der Stack eingerichtet. Danach wird sofort auf ein Modul im fraglichen Adressbereich geprüft. Diese Prüfung sieht so aus, daß ab der Adresse $8004 überprüft wird, ob an den folgenden 5 Speicherzellen die Meldung 'CBM80' steht. Hexadezimal lautet die Folge:

C3 C2 CD 38 30

Ist die Prüfung positiv, so wird an die Adresse verzeigt, die in den ersten beiden Bytes, also $8000 und $8001, abgelegt ist. Das an der angegebenen Adresse liegende selbststartende Programm muß also alle Aufgaben wie Initialisierung der Interrupts, des Bildschirms und des Speichers selbst übernehmen. Dafür können Sie natürlich auf die Systemroutinen zurückgreifen, bei Bedarf aber auch eigene Routinen schreiben.
Wer sich an die Programmierung des Auto-Starts wagen möchte, dem sei wärmstens ein genaues Studium des Betriebssystems an Hand eines Rom-Listings empfohlen.
Verzichten Sie auf den Autostart, so brauchen Sie keine besonderen Vorkehrungen zu treffen. Dann können Sie Ihr Programm ganz einfach in das Eprom brennen. Da die Überprüfung auf Autostart nach dem Einschalten feststellt, daß die geforderte Zeichenkette nicht im Speicher steht, wird der C64 ganz normal initialisiert. Sie können dann das Programm im Eprom mit SYS aktivieren.

Nachdem jetzt Eproms programmiert und am C64 betrieben werden können, stellt sich nur die Frage, was mit Eproms geschieht, die aus irgendeinem Grund gelöscht werden sollen.

Da die im Eprom gespeicherte Information durch Bestrahlung mit ultraviolettem Licht verschwindet, können Sie Ihre Eproms natürlich bis zu einem Besuch in den Alpen oder am Mittelmeer aufbewahren und dort in die Sonne legen. Das ist aber ein etwas langwieriger Prozeß, da selbst bei direkter Sonneneinstrahlung die Information erst nach mehreren Tagen oder sogar Wochen vollständig gelöscht ist. Schneller geht der Löschvorgang mit einer Höhensonne. Die in unserem Versuch verwendete Höhensonne hatte nach spätestens 2 Stunden die Eproms gelöscht. Dabei spielt der Abstand der Eproms zur Höhensonne eine entscheidende Rolle. Je geringer

dieser Abstand ist, um so schneller geht der Löschvorgang. Ganz eilige können sich entweder ein kommerzielles Löschgerät anschaffen oder aber versuchen, ob sie eine UV-strahlende Lampe bekommen können. Eine geeignete Lampe ist zum Beispiel die TUV 6 W von Philips, die Sie sicher in Elekro-Geschäften erhalten können. Bei einem Abstand von ca. lcm sind die Eproms in etwa 30 - 45Minuten gelöscht. Allerdings ist sowohl bei der Löschung mit einer Höhensonne wie auch bei Verwendung einer solchen speziellen UV-Lampe Vorsicht geboten.

Schauen Sie NIE direkt in die eingeschaltete Lampe!

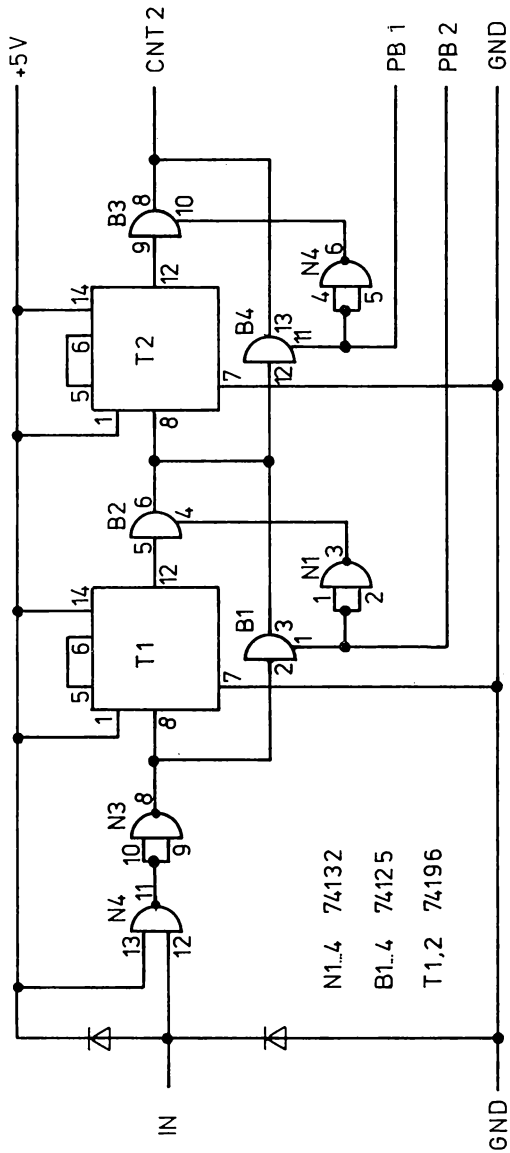
Das erzeugte Licht kann schwerste

Augenschäden hervorrufen.

Auch die Haut ist gegen zu starke Bestrahlung mit UV-Licht allergisch. Die auftretenden Symptome sind Ihnen sicher als Sonnenbrand gut bekannt.
Wenn Sie eine UV-Lampe des genannten oder ähnlichen Typs verwenden wollen, sollten Sie sie unbedingt in ein Gehäuse einbauen. Bedenken Sie dabei auch, daß die Lampe im Betrieb recht heiß wird, wählen Sie das Gehäuse also lieber etwas größer oder sehen Sie Lüftungsschlitze vor. Allerdings sollten Sie bei Anbringung der Lüftungsschlitze darauf achten, daß wohl die heiße Luft, nicht aber auch das UV-Licht nach draußen dringen kann.

9.3 Ein Frequenzzähler bis 30 MHz

Ein Frequenzzähler ist neben Digitalvoltmeter (werden wir im nächsten Kapitel bauen) und Oszilloskop sicher das wertvollste Meßgerät in einem Hobbylabor. Leider sind die Anschaffungskosten für ein solches Gerät trotz drastisch gefallener Halbleiter-Preise immer noch recht hoch. Preiswerter ist die Verwendung des schon vorhandenen C64. Von dieser Überlegung ausgehend ist uns unter Verwendung der besonderen Eigenschaften des C64 eine wirklich universell einsetzbare und sehr preiswerte Zusatzschaltung gelungen, die auch gehobenen Ansprüchen genügen wird. Mit nur vier ICs erhalten Sie einen Frequenzzähler, der in drei Meßbereichen bis maximal 30 MHz mißt. Die Meßbereichsumschaltung wird sogar durch das Programm vorgenommen.
Diese Schaltung benutzt die Möglichkeit, externe Signale mit dem Count-Eingang zu zählen. Als Zähler für diese Impulse arbeitet ein Timer. Diese Aufgabe übernimmt der Timer A der CIA U2.
Um im kleinsten Meßbereich mit einer Genauigkeit von 1 Hertz zu messen, mußte eine Meßzeit von mindestens einer Sekunde gewählt werden. Diese Meßzeit erzeugen die beiden in der CIA U1 integrierten Timer. Das der Timer A der CIA U1 für den Timerinterrupt verwendet wird, soll uns im Moment nicht stören, auf dieses Problem werden wir gleich noch genauer eingehen.
Da ein einzelner Timer nur einen Umfang von 16 Bit hat, wären ohne Vorteiler nur Frequenzen bis maximal 2↑16 = 65536 Hertz zu messen. Aus diesem Grund werden die Unterläufe des Timers A2 bei höheren Frequenzen vom Timer B der CIA 2 gezählt. Dadurch erhalten wir einen 32-Bit-Zähler. Das ergibt einen Zählbereich bis 2↑32 = 4294967296. Der Frequenzzähler könnte also theoretisch bis über 4 Gigahertz messen. Leider setzt uns die Hardware aber Grenzen.



Vorteiler für Frequenzzähler

Wie wir durch Experimente ermittelt haben, beträgt die maximale Eingangsfrequenz des CNT-Eingangs exakt die Hälfte der Taktfrequenz der CIA. Das sind im C64 also 458 KHz. Sollen darüberliegende Frequenzen gemessen werden, so muß ein externer Vorteiler verwendet werden. Genau diese Funktion führt nun unsere kleine Schaltung aus. Sie besteht aus zwei Nand-Gattern mit Schmitt-Trigger-Eingängen zur Impulsformung (74LS132), zwei Dezimalzählern 74LS196 und einem 74LS125. Mit den vier im 74LS125 enthaltenen Tri-State-Buffern und den beiden noch übrigen Schmitt-Triggern werden die Zähler-ICs ein- und ausgeschaltet. Die Steuersignale für die Ein-Ausschaltung liefern die zwei User-Port-Bits PB0 und PB1.

Zur Software ist noch folgendes zu sagen. Der eigentliche Zählvorgang wird mit einer kleinen Maschinensprache-Routine bewerkstelligt, die vom Basic-Programm in den Speicher ab $CF00 gepoked wird. In dieser Routine werden die Timer mit den Startwerten geladen. Dabei werden die Timer A und B der CIA 2 mit dem Wert $FFFF geladen. Wie wir ja im Kapitel 4.0 gesehen haben, können die als Zähler verwendeten Timer nur nach 0 zählen. Jeder auftretende Impuls verringert also den Zählerstand.

Die Timer der CIA 1 werden mit unterschiedlichen Werten initialisiert. Timer A1 wird wie die Timer zuvor mit dem Wert $FFFF geladen. Timer B1 allerdings wird mit dem Wert $0010 geladen. Da Timer A Systemtakte zählt, können wir jetzt ausrechnen, wie lange die beiden Timer A1 und B1 benötigen, bis sie auf 0 heruntergezählt haben.

Timer A benötigt 65536 Taktimpulse bis er auf 0 gezählt hat. Da er im CONTINUOUS-Mode initialisiert wurde, erzeugt er ein Signal beim Unterlauf und beginnt dann von Vorn. Dieser Unterlauf wird im Timer B gezählt. Timer B ist mit dem Wert $0010 = dez. 16 gestartet. Jeweils nach 65536 Impulsen wird also Timer B dekrementiert. Somit sind insgesamt 16 * 65536

Taktimpulse nötig, um den Zähler B auf 0 zu bringen. Das entspricht einer Zeit von etwas mehr als einer Sekunde, genau unsere Meßzeit.
Nachdem im Programm die Timer gestartet wurden, wird im einer Schleife auf das Ende der Meßzeit gewartet. Sobald die Zeit abgelaufen ist, werden die Timer in der CIA2, unsere Impulszähler, gestoppt und die Zählerstände werden in die drei Zero-Page-Adressen $F7, $F8 und $F9 geschrieben. Die obersten 8 Bit des 32-Bit-Timers können aus den oben genannten Gründen ignoriert werden, hier hat sich mit Sicherheit nichts getan.
Jetzt könnte ein aufmerksamer Leser fragen, wieso die Daten erst in die Zero-Page geschrieben werden. Sie könnten ja auch aus den CIA-Registern von Basic aus gePEEKed werden. Grundsätzlich ist diese Möglichkeit schon gegeben. Weil wir aber auch den Timer A1 verwenden, der normalerweise vom System für den Timerinterrupt benutzt wird, müssen wir nach der Messung dem System die Kontrolle über den Timer und die Interrupts zurückgeben. Dies geschieht am einfachsten durch Aufruf der Betriebssystemroutine ab $FDA3. In dieser Routine werden die Timer mit den nach dem Einschalten üblichen Werten geladen. Dadurch ist aber der durch die CNT-Impulse erreichte Zählerstand überschrieben.
Das Basic-Programm startet eine Messung mit SYS 52992. Nach der Meßzeit wird dann der Zählerstand aus den angegebenen Speicherzellen gePEEKed. Die erhaltenen Werte werden nun auf die gemessene Frequenz (enthalten in den Variablen F und F$)umgerechnet und in Form von großen gut sichtbaren Zahlen auf dem Bildschirm dargestellt. Die Zahlen selbst werden als Sprites auf den Bildschirm gebracht.
Liegt der Wert über 320000, so wird der nächsthöhere Meßbereich eingeschaltet. Liegt der Wert unter 30000, dann wird der Meßbereich zurückgeschaltet. Durch diese Art der Hysterese wird das Problem der dauernden Umschaltung

zwischen zwei Meßbereichen an der Umschaltgrenze beseitigt. Ist der gemessene Wert 0, d.h. liegt keine Eingangswechselspannung an, dann wird automatisch in den höchsten Meßbereich geschaltet. Das mag auf den ersten Blick unverständlich erscheinen, hat aber seine Ursache ebenfalls in der Konstruktion des 6526. Bei Frequenzen über 10 MHz kann es nämlich vorkommen, daß gar keine Impulse gezählt werden. Dann könnte das Programm nicht in einen höheren Meßbereich umschalten. Durch die Festlegung auf den höchsten Meßbereich klappt das Herunterschalten aber bestimmmt.
Durch die regelmäßige Initialisierung der CIAs muß jedes mal auch die Steuerinformation für die Vorteiler vor dem SYS neu gePOKEd werden. Dabei gilt die folgende Regel:

PB1	PB0		
0	0	= dez. 0	Frequenz wird direkt gemessen.
0	1	= dez. 1	Frequenz wird durch 10 geteilt.
1	0	= dez. 2	Frequenz wird durch 10 geteilt. Dieser Modus wird nicht verwendet.
1	1	= dez. 3	Frequenz wird durch 100 geteilt.

Insgesamt sind also zwei Steuerleitungen, zwei Leitungen für die Stromversorgung und die Leitung zum CNT-Eingang nötig. Da die Schaltung aus nur vier ICs besteht, ist der Platzbedarf minimal. Somit können Sie die Schaltung gut in ein kleines Röhrchen einbauen und mit einem etwas längeren 5-poligen Kabel mit dem User-Port verbinden. Das Röhrchen ist dann wie ein Tastkopf zu benutzen.
Aber nun das Listing des Programms. Viel Spaß mit diesem wirklich preiswerten und genauen Meßgerät.

```
100 REM ***************************
110 REM *                         *
120 REM *    FREQUENZ ZAEHLER     *
130 REM *                         *
140 REM *      VERSION  1.3       *
150 REM *                         *
160 REM *    ROLF  BRUECKMANN     *
170 REM *    FRIEDHELM  KUECH     *
180 REM ***************************
190 POKE53280,0
200 POKE53281,0
210 PRINTCHR$(5)
220 PRINTCHR$(147):PRINT:PRINT" FREQUENZ ZAEHLER"
230 PRINT : PRINT : PRINT " EINEN MOMENT BITTE"
240 MB = 2
250 ML = 52992  : REM $CF00
260 MH = 53078
270 V  = 53248  : REM VIDEOCONTROLLER
280 Y  =110     : REM Y-POSITION SPRITES
290 D1 = 1024+12*40+18
300 D2 =D1+6    :REM DEZIMAL PUNKT POSITIONEN
310 BUSY=1024+13*40+36
320 FORI=1TO39:BL$=BL$+" ":NEXT
330 F1$="     "
340 FORI=1TO32:F2$=F2$+CHR$(96):NEXT
350 F3$=CHR$(98)+LEFT$(BL$,32)+CHR$(98)
360 LED(0)=1024+10*40+36
370 LED(1)=1024+ 8*40+36
380 LED(2)=1024+ 6*40+36
390 MA(0) =0 : MA(1) =1 : MA(2) = 3
400 POKEV+16,0:  REM  SPRITE 0 CARRY
410 POKEV+23,255:REM  SPRITE Y XPAND
420 POKEV+29,0  :REM NO SPRITE X XPAND
```

```
430 POKEV+28,0  :REM SINGLE COLOUR
440 POKEV,250:POKEV+1,Y:POKEV+2,225:POKEV+3,Y
450 POKEV+4,200:POKEV+5,Y:POKEV+6,175:POKEV+7,Y
460 POKEV+8,150:POKEV+9,Y:POKEV+10,125 :POKEV+11,Y
470 POKEV+12,100:POKEV+13,Y:POKEV+14,75 :POKEV+15,Y
480 FORI=V+39TOV+46
490 :POKEI,1     :REM  SPRITE COLOUR = WHITE
500 NEXT
510 REM    READ & POKE ML PRG
520 FOR I=ML TO MH
530 :READ A
540 :POKEI,A
550 NEXT
560 REM    READ & POKE SPRITES
570 FOR I=12800 TO 13440
580 :READ A
590 :POKEI,A
600 NEXT
610 REM  SET UP NUMBER POINTERS
620 FOR I=0 TO 9
630 :SP(I)=200+I:REM  START
640 NEXT I
650 PRINTCHR$(147):PRINT:PRINT
660 PRINTFl$;CHR$(176);F2$;CHR$(174)
670 PRINTFl$;CHR$(98);"  30 MHZ FREQUENZ ZAEHLER C 64  ";CHR$(98)
680 PRINTFl$;CHR$(171);F2$;CHR$(179)
690 PRINTFl$;CHR$(98);"                              MHZ";CHR$(215);CHR$(98)
700 PRINTFl$;F3$
710 PRINTFl$;CHR$(98);"                              KHZ";CHR$(215);CHR$(98)
720 PRINTFl$;F3$
730 PRINTFl$;CHR$(98);"                               HZ";CHR$(215);CHR$(98)
740 PRINTFl$;F3$
750 PRINTFl$;F3$
760 PRINTFl$;CHR$(98);"                              BSY";CHR$(215);CHR$(98)
```

```
770 PRINTF1$;F3$
780 PRINTF1$;CHR$(173);F2$;CHR$(189)
790 PRINT:PRINT:PRINT:PRINT
800 PRINT"        "CHR$(18)"F1"CHR$(146)" INFO";
810 PRINT"        "CHR$(18)"F7"CHR$(146)" ENDE"
820 GET A$
830 IF A$=CHR$(133) THEN GOTO 1090:REM INFO
840 IF A$=CHR$(136) THEN POKEV+21,0:PRINTCHR$(147):PRINT:PRINT"    ENDE":END
850 POKE BUSY,81:REM  BUSY ON
860 POKE56579,255:POKE56577,MA(MB)
870 SYS 52992:REM MEASUREMENT
880 POKEBUSY,87:REM BUSY OFF
890 A=PEEK(247):B=PEEK(248):C=PEEK(249)
900 ZW=((C*65536+B*256+A+1)-2↑24)*-1
910 K1%=(ZW/100)*6.041:F=ZW-K1%
920 IF F > 320000 THEN IF MB < 2 THEN MB = MB + 1
930 IF F < 30000 THEN IF MB > 0 THEN MB = MB - 1
940 IF F = 0 THEN MB = 2
950 FORI=0TO2:POKE LED(I),87:NEXT:POKE LED(MB),81
960 IFMB=0 OR(MB=2 AND F=0)THEN POKED1,32:POKED2,32:GOTO 990
970 IFMB=1 THEN POKED1,32:POKED2,81
980 IFMB=2 THEN POKED1,81:POKED2,32
990 REM **** FREQUENCY ****
1000 :
1010 F$=STR$(F):LE=LEN(F$)-1:F$=RIGHT$(F$,LE):F$=LEFT$(F$,8)
1020 FOR I=8  TO 1 STEP-1
1030 ::IF I>LE THEN POKE V+21,PEEK(V+21)AND (255-2↑(I-1)):GOTO1060
1040 ::POKE 2040+(LE-I),SP(VAL(MID$(F$,I,1)))
1050 ::POKEV+21,PEEK(V+21)OR2↑(I-1)
1060 NEXT
1070 GOTO 820
1080 REM *** INFORMATION ***
1090 POKEV+21,0:POKE53280,1
1100 PRINTCHR$(147)CHR$(18)"              INFORMATION                "CHR$(146)
```

```
1110 PRINT"   MIT HILFE DIESES PROGRAMMS UND
1120 PRINT"    DER DAZUGEHOERIGEN HARDWARE "
1130 PRINT"  VERWANDELT SICH IHR C64 IN EINEN
1140 PRINT" LEISTUNGSFAEHIGEN FREQUENZZAEHLER
1150 PRINT"  DIE FREQUENZ WIRD AN PIN 6 DES
1160 PRINT"    USER-PORTS (CNT 2) GEMESSEN.":PRINT
1170 PRINT"      DIE OBERE FREQUENZGRENZE
1180 PRINT"       LIEGT BEI ETWA 30 MHZ.
1190 PRINT:PRINT"     DIE EINGANGSSPANNUNG SOLLTE
1200 PRINT:PRINT"              5 VOLT
1210 PRINT:PRINT"   NICHT WESENTLICH UEBERSTEIGEN !
1220 PRINT:PRINTCHR$(18)"             <EINE TASTE>              "CHR$(146)
1230 GETA$:IFA$=""THEN1230
1240 GOTO 650
1250 DATA 120,169,255,141,  4,221,141,  5
1260 DATA 221,141,  6,221,141,  7,221,141
1270 DATA   4,220,141,  5,220,169, 16,141
1280 DATA   6,220,169,  0,141,  7,220,160
1290 DATA  17,162, 89,169, 81,141, 15,221
1300 DATA 169, 49,141, 14,221,140,14,220
1310 DATA 142, 15,220,173,  6,220,208,251
1320 DATA 141, 14,221,141, 15,221,173,  4
1330 DATA 221,133,247,173,  5,221,133,248
1340 DATA 173,  6,221,133,249,173,  7,221
1350 DATA 133,250, 88, 76,163,253,  0, 15
1360 DATA 248,  0, 24, 12,  0, 56, 14,  0,120
1370 DATA  15,  0,120, 15,  0,120, 15,  0,120
1380 DATA  15,  0,120, 15,  0,120, 15,  0,120
1390 DATA  15,  0,120, 15,  0,120, 15,  0,120
1400 DATA  15,  0,120, 15,  0,120, 15,  0,120
1410 DATA  15,  0,120, 15,  0,120, 15,  0, 56
1420 DATA  14,  0, 24, 12,  0, 15,248,  0,  0
1430 DATA   0,224,  0,  1,224,  0,  3,224,  0
1440 DATA   6,224,  0,  0,224,  0,  0,224,  0
```

```
1450 DATA   0,224,  0,  0,224,  0,  0,224,  0
1460 DATA   0,224,  0,  0,224,  0,  0,224,  0
1470 DATA   0,224,  0,  0,224,  0,  0,224,  0
1480 DATA   0,224,  0,  0,224,  0,  0,224,  0
1490 DATA   0,224,  0,  0,224,  0, 15,254,  0
1500 DATA   0, 15,248,  0, 28, 28,  0, 56, 14
1510 DATA   0,120, 15,  0,120, 15,  0,  0, 15
1520 DATA   0,  0, 30,  0,  0, 60,  0,  0,120
1530 DATA   0,  0,240,  0,  1,192,  0,  3,128
1540 DATA   0, 14,  0,  0, 28,  0,  0, 56,  0
1550 DATA   0,120, 12,  0,112, 30,  0,127,255
1560 DATA   0,126,255,  0,120, 62,  0,112, 12
1570 DATA   0,  0, 15,248,  0, 28, 28,  0, 56
1580 DATA  14,  0,120, 15,  0,120, 15,  0,  0
1590 DATA  15,  0,  0, 15,  0,  0, 14,  0,  0
1600 DATA  12,  0,  1,248,  0,  1,248,  0,  0
1610 DATA  12,  0,  0, 14,  0,  0, 14,  0,  0
1620 DATA  15,  0,  0, 15,  0,120, 15,  0,120
1630 DATA  15,  0, 56, 14,  0, 28, 28,  0, 15
1640 DATA 248,  0,  0,  0,224,  0,  1,224,  0
1650 DATA   3,224,  0,  7,224,  0, 14,224,  0
1660 DATA  28,224,  0, 56,224,  0,112,224,  0
1670 DATA 112,224,  0,224,224,  0,224,224,  0
1680 DATA 224,224,  0,255,254,  0,  0,224,  0
1690 DATA   0,224,  0,  0,224,  0,  0,224,  0
1700 DATA   0,224,  0,  0,224,  0,  0,224,  0
1710 DATA  15,254,  0,  0,127,255,  0,120,  0
1720 DATA   0,120,  0,  0,120,  0,  0,120,  0
1730 DATA   0,120,  0,  0,120,  0,  0,127,240
1740 DATA   0,127,252,  0,  0, 62,  0,  0, 30
1750 DATA   0,  0, 15,  0,  0, 15,  0,  0,  7
1760 DATA   0,  0,  7,  0,  0,  7,  0,112,  7
1770 DATA   0,120, 15,  0, 56, 14,  0, 28, 28
1780 DATA   0, 15,248,  0,  0, 15,252,  0, 28
```

```
1790 DATA  14,  0, 56, 15,  0,112,  0,  0,112
1800 DATA   0,  0,112,  0,  0,112,  0,  0,112
1810 DATA   0,  0,119,240,  0,124, 60,  0,124
1820 DATA  30,  0,120, 15,  0,112, 15,  0,112
1830 DATA   7,  0,112,  7,  0,112,  7,  0,112
1840 DATA   7,  0,120, 15,  0, 56, 14,  0, 28
1850 DATA  28,  0, 15,248,  0,  0,127,255,  0
1860 DATA 120, 15,  0,120, 15,  0,120, 15,  0
1870 DATA   0, 15,  0,  0, 14,  0,  0, 30,  0
1880 DATA   0, 60,  0,  0,120,  0,  0,112,  0
1890 DATA   0,224,  0,  0,224,  0,  1,192,  0
1900 DATA   1,192,  0,  1,192,  0,  1,192,  0
1910 DATA   1,192,  0,  1,192,  0,  1,192,  0
1920 DATA   1,192,  0,  1,192,  0,0, 15,248
1930 DATA   0, 28, 28,  0, 56, 14,  0,112,  7
1940 DATA   0,112,  7,  0,112,  7,  0,112,  7
1950 DATA   0, 56, 14,  0, 28, 28,  0, 15,248
1960 DATA   0, 15,248,  0, 28, 28,  0, 56, 14
1970 DATA   0,112,  6,  0,112,  7,  0,112,  7
1980 DATA   0,112,  7,  0,120, 15,  0, 56, 14
1990 DATA   0, 28, 28,  0, 15,248,  0,  0, 15
2000 DATA 252,  0, 28, 14,  0, 56, 15,  0,112
2010 DATA   7,  0,112,  7,  0,112,  7,  0,112
2020 DATA   7,  0,112,  7,  0, 56,  7,  0, 28
2030 DATA  15,  0, 15,255,  0,  0,  7,  0,  0
2040 DATA   7,  0,  0,  7,  0,  0,  7,  0,  0
2050 DATA   7,  0,  0,  7,  0,120, 15,  0, 56
2060 DATA  14,  0, 28, 28,  0, 15,248,  0,  0
2070 DATA  92
```

9.4 Auto-DVM

Das im Kapitel 7.4 beschriebene IC CA 3162 eignet sich ausgezeichnet zum Bau eines Digital-Voltmeters. Das Experimentieren reizte uns zur Entwicklung eines Voltmeters mit automatischer Meßbereichswahl.

Weit über 90 % aller mit einem Meßgerät ausgeführten Messungen sind Gleichspannungsmessungen. Darum und um die Kosten für unsere Bauanleitung gering zu halten, haben wir uns auf die Gleichspannungsmessung beschränkt. Mit diesem Voltmeter können Gleichspannungen zwischen -10 Volt und +100 Volt gemessen werden.

Wie auch beim zuvor beschriebenen Frequenzzähler ist die Hardware so klein, daß sie ohne Probleme in einem kleinen Tastkopfgehäuse untergebracht werden kann. Allerdings reicht bei dieser Schaltung kein 5-poliges Kabel für den Anschluß an den User-Port. Wir benötigen ein 10-poliges Kabel.

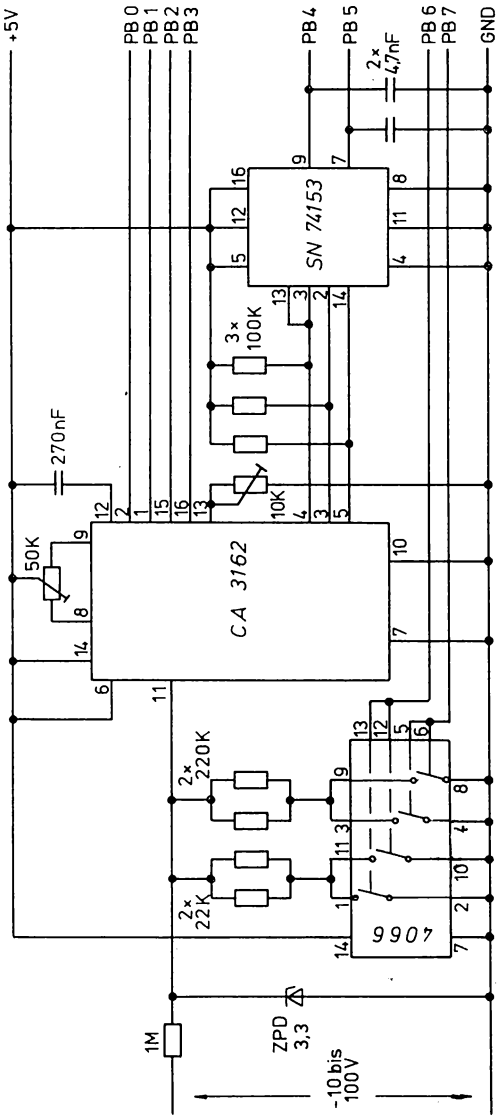
Da die grundsätzliche Arbeitsweise des CA 3162 bereits im Kapitel 7.4 beschrieben sind, wollen wir uns auf die Besonderheiten dieser Schaltung beschränken.

Die vom Voltmeter gelieferten Daten werden über die vier User-Port-Bits PB0 bis PB3 an den C64 übergeben. Die drei Digit- oder Stellenleitungen des DVM-ICs werden allerdings nicht direkt an den User-Port geführt. Da für die Meßbereichswahl bei drei Meßbereichen mindestens 2 Bit benötigt werden, haben wir die Digit-Signale über das IC 74LS153 gemultiplext. Damit benötigen wir auch für die Stelleninformation nur 2 Leitungen. Das ergibt insgesamt 8 Leitungen, genau die vom User-Port zur Verfügung gestellte Anzahl von Bits.

Die Aufteilung der oberen vier Bits wurde nun so vorgenommen, daß die Bits PB4 und PB5 die gemultiplexte Digitinformation enthalten und über die beiden verbleibenden Bits die Meßbereiche geschaltet werden.

Die Umschaltung wird durch CMOS-Analogschalter vom Typ 4066 vorgenommen. Diese Schalter weisen im durchgeschalteten Zustand einen Widerstand von etwa 200 Ohm auf. Sind die Schalter gesperrt, dann liegt ihr Widerstand bei mehreren 100 MOhm. Das Meßergebnis wird also durch die Schalter nur minimal beeinflußt.

Ein Tip für Leute, die eine möglichst hohe Genauigkeit wünschen. Ersetzen Sie die Analogschalter durch Reed-Relais. Diese haben im eingeschalteten Zustand einen Widerstand von wenigen Milliohm. Allerdings müsste die Ansteuerschaltung dann natürlich modifiziert werden. Geeignete Schaltungen wurden ja bereits besprochen. Auch sollten beim Einsatz von Relais mindestens 1%ige Widerstände verwendet werden, sonst verschenken Sie die zusätzliche Genauigkeit.



Digital-Voltmeter mit automatischer Meßbereichsumschaltung

```
100 REM ***************************
110 REM *                         *
120 REM *    AUTO DIGITAL VOM     *
130 REM *                         *
140 REM *     VERSION  1.3.2      *
150 REM *                         *
160 REM *    ROLF  BRUECKMANN     *
170 REM *    FRIEDHELM  KUECH     *
180 REM ***************************
190 PRINTCHR$(147):PRINT:PRINT"       AUTOMATIC DIGITAL VOLTMETER"
200 PRINT:PRINT:PRINT"        EINEN MOMENT BITTE
210 ME(1)= 0 :ME(2)= 128 :ME(3)= 64
220 ML=52992
230 MH=53032
240 V=53248
250 Y=110:REM Y-POSITION SPRITES
260 BUSY=1024+13*40+36
270 Z$="000"
280 FORI=1TO32:F0$=F0$+" ":F2$=F2$+CHR$(96):NEXT
290 F1$="    "
300 F3$=CHR$(98)+F0$+CHR$(98)
310 F4$=CHR$(19)
320 FORI=1TO12:F4$=F4$+CHR$(17):NEXT
330 FORI=1TO22:F4$=F4$+CHR$(29):NEXT
340 F5$=CHR$(19)
350 FORI=1TO10:F5$=F5$+CHR$(17):NEXT
360 FORI=1TO11:F5$=F5$+CHR$(29):NEXT
370 F6$=CHR$(157)+CHR$(157)+CHR$(157)+CHR$(145)+CHR$(145)+CHR$(32)+CHR$(157)
380 F6$=F6$+CHR$(17)+CHR$(32)+CHR$(17)+CHR$(17)+CHR$(157)+CHR$(32)+CHR$(17)
390 F6$=F6$+CHR$(157)+CHR$(32)
400 POKE56577,0:ME=1:REM  LOWEST RANGE
410 POKE53281,0:POKE53280,1
420 POKEV+16,0:REM  SPRITE 0 CARRY
```

```
430 POKEV+23,255:REM  SPRITE Y XPAND
440 POKEV+29,0  :REM NO SPRITE X XPAND
450 POKEV+28,0  :REM SINGLE COLOUR
460 POKEV,225:POKEV+1,Y:POKEV+2,200:POKEV+3,Y
470 POKEV+4,175:POKEV+5,Y:POKEV+6,150:POKEV+7,Y
480 POKEV+8,125:POKEV+9,Y:POKEV+10,100 :POKEV+11,Y
490 POKEV+12,100:POKEV+13,Y:POKEV+14,75:POKEV+15,Y
500 FORI=V+39TOV+46
510 :POKEI,1     :REM  SPRITE COLOUR = WHITE
520 NEXT
530 REM    READ & POKE ML PRG
540 FOR I=ML TO MH
550 :READA:CS=CS+A
560 :POKEI,A
570 NEXT
580 REM    READ & POKE SPRITES
590 FOR I=12800 TO 13440
600 :READA:CS=CS+A
610 :POKEI,A
620 NEXT
630 IFCS<>37591THENPRINT"FEHLER IN DEN DATAS!":END
640 POKEV+21,1  :REM SPRITE 0 ON
650 POKE2040,200:REM FIRST DIGIT = 0
660 REM  SET UP NUMBER POINTERS
670 FOR I=0 TO 9
680 ::SP(I)=200+I:REM  START
690 NEXT I
700 PRINTCHR$(147),CHR$(5):PRINT:PRINT
710 PRINTF1$;CHR$(176);F2$;CHR$(174)
720 PRINTF1$;CHR$(98);"  AUTO RANGE DIGITAL VOLTMETER  ";CHR$(98)
730 PRINTF1$;CHR$(171);F2$;CHR$(179)
740 PRINTF1$;F3$
750 PRINTF1$;F3$
760 PRINTF1$;CHR$(98);"                                V "CHR$(98)
```

```
770 PRINTF1$;F3$
780 PRINTF1$;CHR$(98);"                                MV "CHR$(98)
790 PRINTF1$;F3$
800 PRINTF1$;F3$
810 PRINTF1$;CHR$(98);"                               BSY "CHR$(98)
820 PRINTF1$;F3$
830 PRINTF1$;CHR$(173);F2$;CHR$(189)
840 PRINT:PRINT:PRINT:PRINT
850 PRINT"      "CHR$(18);"F1";CHR$(146);" INFO";
860 PRINT"      "CHR$(18);"F7";CHR$(146);" ENDE"
870 GETA$
880 IFA$=CHR$(133)THENGOTO1320:REM INFO
890 IFA$=CHR$(136)THENPOKEV+21,0:PRINTCHR$(147):PRINTCHR$(5);"ENDE":END
900 POKEBUSY,81:REM  BUSY ON
910 SYS52992:REM MEASUREMENT
920 POKEBUSY,87:REM BUSY OFF
930 Z$=""
940 :
950 FORI=0 TO 2
960 :GOSUB 1480
970 :Z$=Z$+RIGHT$(A$,1)
980 NEXTI
990 IF Z$="---" THEN ME=ME+1:GOTO 1030
1000 IF Z$="↑↑↑" THEN ME=ME+1:GOTO 1030
1010 IF VAL(Z$)>0 THEN IF VAL(Z$)<100 THEN ME=ME-1
1020 IF VAL(Z$)<0 THEN IF VAL(Z$)>-10 THEN ME=ME-1
1030 IF ME<1 THEN ME=1
1040 IF ME>3 THEN ME=3: GOSUB 1270:GOTO 870
1050 POKE56577,ME(ME):REM RANGE
1060 PRINTF4$;
1070 IFME=1THENPRINT"       "
1080 IFME=2THENPRINTCHR$(5);CHR$(113);"      "
1090 IFME=3THENPRINTCHR$(5);"   "CHR$(113);"   "
1100 IF ME=1 THEN POKE1024+400+33,81:POKE1024+320+33,32:GOTO 1120
```

```
1110 POKE1024+320+33,81:POKE1024+400+33,32
1120 IF LEFT$(Z$,1)="-" THEN Z$=RIGHT$(Z$,2):GOSUB 1210:GOTO 1140
1130 GOSUB 1240:REM  PLUS SIGN
1140 LE=LEN(Z$)
1150 FOR I=3 TO 1 STEP -1
1160 ::IF I>LE THEN POKE V+21,PEEK(V+21)AND (255-2↑(I-1)):NEXT
1170 ::POKE2040+(LE-I),SP(VAL(MID$(Z$,I,1)))
1180 ::POKEV+21,PEEK(V+21)OR2↑(I-1)
1190 NEXT
1200 GOTO 870:REM  MAIN LOOP
1210 REM **** PRINT MINUS SIGN ****
1220 PRINTF5$;CHR$(31);CHR$(18);"     "CHR$(146);F6$
1230 RETURN
1240 REM **** PRINT PLUS  SIGN ****
1250 PRINTF5$;CHR$(28);CHR$(18);"     ";F6$
1260 RETURN
1270 REM **** OVERFLOW ****
1280 REM FLASHES BACKGROUND, SHOWS MESSAGE
1290 PRINTF5$;CHR$(29);"OVER"
1300 POKE53281,1:FORP=1TO100:NEXT:POKE53281,0:RETURN
1310 REM *** INFORMATION ***
1320 POKEV+21,0:POKE53280,1
1330 PRINTCHR$(147);CHR$(5);CHR$(18)"            INFORMATION
1340 PRINT" MIT DER ENTSPRECHENDEN HARDWARE WIRD
1350 PRINT"  IHR COMMODORE 64 ZU EINEM LOW-COST"
1360 PRINT:PRINT"   ";CHR$(18);" AUTOMATIC DIGITAL VOLTMETER ";CHR$(146)
1370 PRINT:PRINT
1380 PRINT"  DIE MAXIMALE EINGANSSPANNUNG SOLLTE
1390 PRINT:PRINT"  ";CHR$(18)" NIE 200 VOLTS DC UEBERSTEIGEN ! "CHR$(146)
1400 PRINT:PRINT:PRINT"     SOLLTE DIE ANZEIGE ZU FLIMMERN
1410 PRINT"       BEGINNEN, SO HABEN SIE EINE
1420 PRINT:PRINT"  GEFAEHRLICH HOHE EINGANGSSPANNUNG
1430 PRINT:PRINT"     AN DEN EINGANGSBUCHSEN!!!
1440 PRINT:PRINT
```

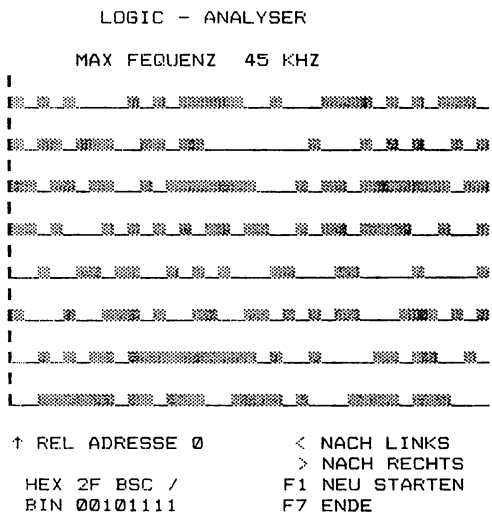
```
1450 PRINTCHR$(18)"              < EINE TASTE >              "CHR$(146)
1460 GETA$:IFA$=""THEN1460
1470 GOTO700
1480 REM **** EVALUATE STRING ****
1490 A=PEEK(249+I)
1500 IF A=10 THEN A$="-":RETURN
1510 IF A=11 THEN A$="↑":RETURN
1520 A$=STR$(A)
1530 RETURN
1540 REM  ****  ML PROGRAM ****
1550 DATA 120,169,000,133,255,162,249,169
1560 DATA 192,141,003,221,173,001,221,168
1570 DATA 041,048,197,255,208,246,152,041
1580 DATA 015,149,000,232,165,255,024,105
1590 DATA 016,201,048,133,255,208,229,088,096
1600 DATA   0, 15
1610 DATA 248,  0, 24, 12,  0, 56, 14,  0,120
1620 DATA  15,  0,120, 15,  0,120, 15,  0,120
1630 DATA  15,  0,120, 15,  0,120, 15,  0,120
1640 DATA  15,  0,120, 15,  0,120, 15,  0,120
1650 DATA  15,  0,120, 15,  0,120, 15,  0,120
1660 DATA  15,  0,120, 15,  0,120, 15,  0, 56
1670 DATA  14,  0, 24, 12,  0, 15,248,  0,  0
1680 DATA   0,224,  0,  1,224,  0,  3,224,  0
1690 DATA   6,224,  0,  0,224,  0,  0,224,  0
1700 DATA   0,224,  0,  0,224,  0,  0,224,  0
1710 DATA   0,224,  0,  0,224,  0,  0,224,  0
1720 DATA   0,224,  0,  0,224,  0,  0,224,  0
1730 DATA   0,224,  0,  0,224,  0,  0,224,  0
1740 DATA   0,224,  0,  0,224,  0, 15,254,  0
1750 DATA   0, 15,248,  0, 28, 28,  0, 56, 14
1760 DATA   0,120, 15,  0,120, 15,  0,  0, 15
1770 DATA   0,  0, 30,  0,  0, 60,  0,  0,120
1780 DATA   0,  0,240,  0,  1,192,  0,  3,128
```

```
1790 DATA   0, 14,  0,  0, 28,  0,  0, 56,  0
1800 DATA   0,120, 12,  0,112, 30,  0,127,255
1810 DATA   0,126,255,  0,120, 62,  0,112, 12
1820 DATA   0,  0, 15,248,  0, 28, 28,  0, 56
1830 DATA  14,  0,120, 15,  0,120, 15,  0,  0
1840 DATA  15,  0,  0, 15,  0,  0, 14,  0,  0
1850 DATA  12,  0,  1,248,  0,  1,248,  0,  0
1860 DATA  12,  0,  0, 14,  0,  0, 14,  0,  0
1870 DATA  15,  0,  0, 15,  0,120, 15,  0,120
1880 DATA  15,  0, 56, 14,  0, 28, 28,  0, 15
1890 DATA 248,  0,  0,  0,224,  0,  1,224,  0
1900 DATA   3,224,  0,  7,224,  0, 14,224,  0
1910 DATA  28,224,  0, 56,224,  0,112,224,  0
1920 DATA 112,224,  0,224,224,  0,224,224,  0
1930 DATA 224,224,  0,255,254,  0,  0,224,  0
1940 DATA   0,224,  0,  0,224,  0,  0,224,  0
1950 DATA   0,224,  0,  0,224,  0,  0,224,  0
1960 DATA  15,254,  0,  0,127,255,  0,120,  0
1970 DATA   0,120,  0,  0,120,  0,  0,120,  0
1980 DATA   0,120,  0,  0,120,  0,  0,127,240
1990 DATA   0,127,252,  0,  0, 62,  0,  0, 30
2000 DATA   0,  0, 15,  0,  0, 15,  0,  0,  7
2010 DATA   0,  0,  7,  0,  0,  7,  0,112,  7
2020 DATA   0,120, 15,  0, 56, 14,  0, 28, 28
2030 DATA   0, 15,248,  0,  0, 15,252,  0, 28
2040 DATA  14,  0, 56, 15,  0,112,  0,  0,112
2050 DATA   0,  0,112,  0,  0,112,  0,  0,112
2060 DATA   0,  0,119,240,  0,124, 60,  0,124
2070 DATA  30,  0,120, 15,  0,112, 15,  0,112
2080 DATA   7,  0,112,  7,  0,112,  7,  0,112
2090 DATA   7,  0,120, 15,  0, 56, 14,  0, 28
2100 DATA  28,  0, 15,248,  0,  0,127,255,  0
2110 DATA 120, 15,  0,120, 15,  0,120, 15,  0
2120 DATA   0, 15,  0,  0, 14,  0,  0, 30,  0
```

```
2130 DATA   0, 60,  0,  0,120,  0,  0,112,  0
2140 DATA   0,224,  0,  0,224,  0,  1,192,  0
2150 DATA   1,192,  0,  1,192,  0,  1,192,  0
2160 DATA   1,192,  0,  1,192,  0,  1,192,  0
2170 DATA   1,192,  0,  1,192,  0,0, 15,248
2180 DATA   0, 28, 28,  0, 56, 14,  0,112,  7
2190 DATA   0,112,  7,  0,112,  7,  0,112,  7
2200 DATA   0, 56, 14,  0, 28, 28,  0, 15,248
2210 DATA   0, 15,248,  0, 28, 28,  0, 56, 14
2220 DATA   0,112,  6,  0,112,  7,  0,112,  7
2230 DATA   0,112,  7,  0,120, 15,  0, 56, 14
2240 DATA   0, 28, 28,  0, 15,248,  0,  0, 15
2250 DATA 252,  0, 28, 14,  0, 56, 15,  0,112
2260 DATA   7,  0,112,  7,  0,112,  7,  0,112
2270 DATA   7,  0,112,  7,  0, 56,  7,  0, 28
2280 DATA  15,  0, 15,255,  0,  0,  7,  0,  0
2290 DATA   7,  0,  0,  7,  0,  0,  7,  0,  0
2300 DATA   7,  0,  0,  7,  0,120, 15,  0, 56
2310 DATA  14,  0, 28, 28,  0, 15,248,  0,  0
2320 DATA  92
```

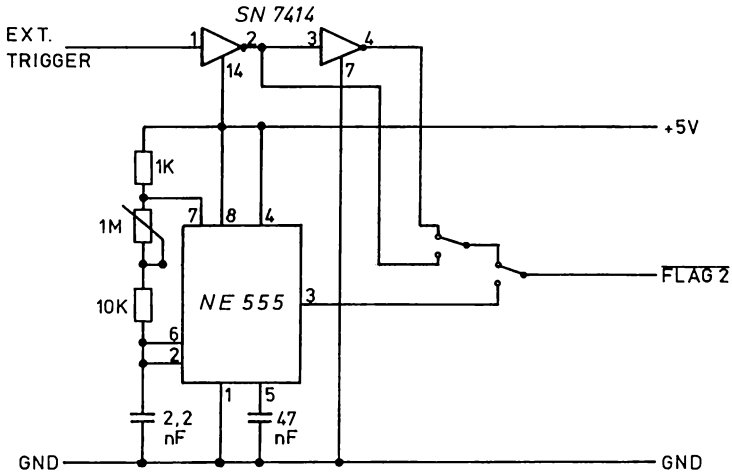
9.5 Logic-Analyzer

Logic-Analyser haben ihren Einsatz in der Entwicklung und der Reparatur hochkomplexer Digitalschaltungen. Die Funktion dieser Geräte ist am ehesten mit einem Viel-Kanal-Oszillografen zu beschreiben. Ein solcher Analyzer hat in der Regel mindestens 8 Dateneingänge. Über diese Eingänge werden für einen bestimmten Zeitraum Daten in den Analyzer gegeben und dort gespeichert. Anschließend können die Daten in Ruhe ausgewertet werden.
Das Bild zeigt einen Bildschirmabdruck mit zufälligen Anzeigewerten.



Der Unterschied zum Oszillografen mit mehreren Eingängen besteht also in der Speicherung der Daten. Der Oszillograf zeigt die an den Eingängen liegenden Daten oder Signale ja sofort an. Dafür sind sie aber auch in Null Komma Nichts

verschwunden. Wer einmal mit einem Oszillografen die Signale auf dem Datenbus eines Prozessors gemessen hat, weiß um die Schwierigkeiten der Auswertung.
Leider ist unser Logic-Analyzer nicht in der Lage, die Signale auf dem Datenbus des C64 zu messen. Da der größte Teil der Arbeit von einem Programm übernommen wird, erreichen wir nur wesentlich geringere Arbeitsfrequenzen als z.B. die Taktfrequenz des C64.
Die maximale Arbeitsfrequenz liegt bei etwa 40 KHz. Folgen die Eingangssignale mit einer größeren Frequenz aufeinander, so versagt unser Prinzip. In diesem Fall werden einzelne Informationen einfach ignoriert. Das erzeugte Bild, auch Timingdiagramm genannt, enthält dann nicht mehr alle aufgetretenen Signale.
Diese Geschwindigkeitsgrenze tut der Nützlichkeit unseres Analyzers aber keinen entscheidenden Abbruch. Auch mit der relativ niedrigen Grenzfrequenz läßt sich das Gerät durchaus für verschiedene Anwendungen einsetzen.
Aus den eben schon erwähnten Geschwindigkeitsgründen wurde das Programm zu einem großen Teil in Maschinensprache geschrieben.
Das nachstehend abgedruckte Assemblerlisting ist kommentiert, so daß an dieser Stelle eine grobe Funktionsbeschreibung genügen soll.
Die eigentliche Messung wird mit einer Schleife vorgenommen. Sobald am -FLAG-Eingang eine negative Flanke aufgetreten ist, werden die am User-Port liegenden Daten in den Speicher gelesen. Auf diese Weise werden nacheinander 256 Bytes von den Datenleitungen in den Speicher gebracht.
Für die Dauer der Messung wir der Bildschirm abgeschaltet. Ohne diese Maßnahme würde der Video-Controller beim Beginn jeder Bildschirmzeile für einen kurzen Zeitraum den Prozessor unterbrechen. Diese Unterbrechungen liegen im Bereich von ca. 40 Microsekunden, etwa dem doppelten unserer



Programmschleife bei höchster Meßrate. Das könnte aber zu einem Datenverlust führen.
Nachdem die 256 Bytes im Puffer untergebracht worden sind, werden die ersten 37 Bytes auf dem Bildschirm angezeigt.
Danach springt das Programm in das aufrufende Basicprogramm zurück.
Zur Anzeige der übrigen Werte hat das Programm die beiden Einsprungspunkte LEFT und RIGHT. Je nach gewähltem Einsprung wird dann die Anzeige um ein Byte verschoben. Auf diese Weise können alle 256 Bytes als Timingdiagramm angezeigt werden.
Zusätzlich werden noch die hexadezimalen und binären Werte sowie der Bildschirmcode der ganz links in der Anzeige stehenden Meßdaten ausgegeben.

```
DD00          CIA2      =    $DD00
DD03          DDRB      =    CIA2 + 3
DD01          PORTB     =    CIA2 + 1
DD0D          ICR       =    CIA2 + 13
D000          VIC       =    $D000
D011          STREG     =    VIC + 17
00F7          POINTER   =    $F7
00F9          ZEILE     =    $F9
0001          FARBE     =    1        ;WEISS
D800          COLRAM    =    $D800
C000                    *=   $C000
C000 78       READLOOP  SEI              ;INTERRUPTS SPERREN
C001 AD 11 D0           LDA  STREG
C004 29 EF              AND  #%11101111  ;BILD AUSSCHALTEN
C006 8D 11 D0           STA  STREG
C009 A2 00              LDX  #0          ;SCHLEIFENZAEHLER
C00B A9 7F              LDA  #$7F        ;EVTL. INTERRUPT LOESCHEN
C00D 8D 0D DD           STA  ICR
C010 A9 10    LOOP      LDA  #%00010000  ;-FLAG-BIT IM ICR
C012 2C 0D DD LOOP1     BIT  ICR
C015 F0 FB              BEQ  LOOP1       ;SIGNAL AUFGETRETEN?
C017 AD 01 DD           LDA  PORTB       ;WENN JA, PORT LESEN
C01A 9D 00 C1           STA  BUFFER,X    ;UND WERT IN BUFFER
C01D E8                 INX              ;UND DAS 256 MAL
C01E D0 F0              BNE  LOOP
C020 AD 11 D0           LDA  STREG
C023 09 10              ORA  #%00010000  ;BILDSCHIRM WIEDER AN
C025 8D 11 D0           STA  STREG
C028 A9 01    FRFILL    LDA  #FARBE      ;CODE FÜR WEISS
C02C 99 00 D8 FRLOOP    STA  COLRAM,Y
C02F 99 00 D9           STA  COLRAM+$100,Y   ; UND DAS FARB-RAM
C032 99 00 DA           STA  COLRAM+$200,Y   ; FÜLLEN
C035 99 00 DB           STA  COLRAM+$300,Y
```

```
C038 C8                INY
C039 D0 F1             BNE  FRLOOP
C03B A9 00             LDA  #<BUFFER     ;ADRESSE DES BUFFERS IN
C03D 85 F7             STA  POINTER      ;DIE ZERO-PAGE SCHREIBEN
C03F A9 C1             LDA  #>BUFFER
C041 85 F8             STA  POINTER+1
C043 58                CLI               ;INTERRUPTS WIEDER ERLAUBEN
C044 4C 4E C0          JMP  OUT          ;ERSTE 'SEITE' ANZEIGEN
C047 C6 F7     LEFT    DEC  POINTER      ;EINSPRUNG FÜR <
C049 4C 4E C0          JMP  OUT
C04C E6 F7     RIGHT   INC  POINTER      ;EINSPRUNG FÜR >
C04E           ; AUSGABE EINES BILDSCHIRMS
C04E A0 24     OUT     LDY  #36          ;ANZAHL DER STELLEN
C050 A2 07     OUTL1   LDX  #7           ;ANZ DER BITS = TIMINGDIAGRAMME
C052 B1 F7             LDA  (POINTER),Y  ;BYTE FUER DIE ANZEIGE HOLEN
C054 6A        OUTLOOP ROR               ;ANZUZEIGENDES BIT IN CARRY
C055 48                PHA
C056 B0 04             BCS  EINS         ;CARRY GESETZT = BIT GESETZT
C058 A9 64             LDA  #100         ;HIER JE NACHDEM, OB BIT
C05A D0 02             BNE  AUS          ;GESETZT ODER GELÖSCHT
C05C A9 66     EINS    LDA  #102         ;ENTSPRECHENDES GRAPHIKZEICHEN
C05E 48        AUS     PHA
C05F 8A                TXA
C060 48                PHA               ;BERECHNUNG DER BILDSCHIRM-
C061 0A                ASL               ;POSITION FÜR DIE AUSGABE
C062 AA                TAX
C063 BD B6 C0          LDA  TABLE,X
C066 85 F9             STA  ZEILE
C068 BD B7 C0          LDA  TABLE+1,X
C06B 85 FA             STA  ZEILE+1
C06D 68                PLA
C06E AA                TAX
C06F 68                PLA
C070 91 F9             STA  (ZEILE),Y
```

```
C072 68              PLA
C073 CA              DEX             ;NAECHSTES BIT
C074 10 DE           BPL  OUTLOOP
C076 88              DEY             ;NAECHSTES BYTE
C077 10 D7           BPL  OUTL1
C079 A0 00           LDY  #0
C07B B1 F7           LDA  (POINTER),Y ;BYTE VON REL ADRESSE LADEN
C07D 8D 7F 07        STA  1919       ;ALS BSC AUSGEBEN
C080 48              PHA
C081 6A              ROR             ;AUSGABE ALS HEXWERT
C082 6A              ROR
C083 6A              ROR
C084 6A              ROR
C085 29 0F           AND  #$0F
C087 18              CLC
C088 69 30           ADC  #48
C08A C9 3A           CMP  #58
C08C 90 03           BCC  HEXOUT
C08E 18              CLC
C08F E9 38           SBC  #56
C091 8D 78 07 HEXOUT STA  1912
C094 68              PLA
C095 48              PHA
C096 29 0F           AND  #$0F
C098 18              CLC
C099 69 30           ADC  #48
C09B C9 3A           CMP  #58
C09D 90 03           BCC  HEXOUT1
C09F 18              CLC
C0A0 E9 38           SBC  #56
C0A2 8D 79 07 HEXOUT1 STA 1913
C0A5 68              PLA
C0A6 A0 07           LDY  #7         ;AUSGABE ALS BINÄRWERT
C0A8 6A       BTL    ROR
```

```
C0A9 48              PHA
C0AA A9 30           LDA  #48
C0AC 69 00           ADC  #0
C0AE 99 A0 07        STA  1952,Y
C0B1 68              PLA
C0B2 88              DEY
C0B3 10 F3           BPL  BTL
C0B5 60              RTS             ;UND ZURÜCK INS BASIC-PRG
C0B6 D3 06    TABLE  .WORD1024 +(18*40)+3   ;
C0B8 83 06           .WORD1024 +(16*40)+3   ;IN DIESER TABELLE
C0BA 33 06           .WORD1024 +(14*40)+3   ;SIND DIE ACHT
C0BC E3 05           .WORD1024 +(12*40)+3   ;ADRESSEN DER ANFANGS-
C0BE 93 05           .WORD1024 +(10*40)+3   ;PUNKTE DER BILDSCHIRM-
C0C0 43 05           .WORD1024 +( 8*40)+3   ;AUSGABE ABGELEGT
C0C2 F3 04           .WORD1024 +( 6*40)+3
C0C4 A3 04           .WORD1024 +( 4*40)+3
C100                 *=   $C100      ;DER BUFFER FÜR DIE 256 BYTES
C100         BUFFER  =    *          ;LIEGT VON $C100 - $C1FF
```

Dieses Programm haben wir in ein kleines Basic-Programm eingebunden. In den Datazeilen steht das Maschinensprache-Programm, das in der bekannten Art und Weise in den Speicher ab $C000 gePOKEd wird. In diesem Programm sind keine Besonderheiten enthalten. Das Verständnis sollte einem geübten Basic-Programmierer nicht schwer fallen.

Die Handhabung des Programms ist recht einfach. Nach dem Start des Programms wird erst einmal die Bildschirmmaske erzeugt. Jetzt kann durch Druck auf die Taste F1 ein Meßvorgang eingeleitet werden.

Nachdem am -FLAG-Eingang 256 Impulse aufgetreten sind, wird das Diagramm der ersten 37 Bytes angezeigt. Mit den Tasten < und > (mit oder ohne SHIFT) wird die Anzeige dann nach links oder rechts gescrollt. Durch erneuten Druck auf F1 können Sie eine neue Messung vornehmen, F7 beendet das Programm.

```
100 POKE650,128 : POKE 247,0
110 GOSUB450
120 FORI=1TO16:CC$=CC$+CHR$(29):NEXT
130 FORI=1TO20:CB$=CB$+CHR$(17):NEXT:CB$=CHR$(19)+CB$
140 LM$="  "+CHR$(182):CL$=CHR$(29)
150 FORI=0TO40:POKE49664+I,0:NEXT
160 PRINTCHR$(144);CHR$(147);"          LOGIC - ANALYSER
170 PRINT:PRINT"        MAX FREQUENZ  45 KHZ
180 IF F THEN SYS49152
190 PRINTLM$  :PRINT "D0";CHR$(182)
200 PRINTLM$  :PRINT "D1";CHR$(182)
210 PRINTLM$  :PRINT "D2";CHR$(182)
220 PRINTLM$  :PRINT "D3";CHR$(182)
230 PRINTLM$  :PRINT "D4";CHR$(182)
240 PRINTLM$  :PRINT "D5";CHR$(182)
250 PRINTLM$  :PRINT "D6";CHR$(182)
260 PRINTLM$:PRINT"D7";CHR$(182)
270 PRINT:PRINT"   ↑ REL ADRESSE ..       < NACH LINKS"
280 PRINT"                            > NACH RECHTS"
290 PRINT"    HEX";
300 PRINTCL$;CL$;CL$;CL$;"BSC";
310 FORI=1TO10:PRINTCL$;:NEXT:PRINT"F1 NEU STARTEN"
320 PRINT"    BIN";
330 FORI=1TO17:PRINTCL$;:NEXT:PRINT"F7 ENDE"
340 PRINTCB$;
350 AD = PEEK(247):AD$=STR$(AD)
360 AD$=CHR$(5)+AD$+"   "+CHR$(144)
370 PRINTCC$;AD$
380 PRINTCHR$(145);
390 GETA$:IFA$=""THEN390
400 IFFTHENIFA$=CHR$(60)ORA$=CHR$(44)THENSYS49223:GOTO350
410 IFFTHENIFA$=CHR$(62)ORA$=CHR$(46)THENSYS49228:GOTO350
420 IF A$=CHR$(133)THENF=1:  GOTO 150
430 IFA$=CHR$(136)THENPRINTCHR$(147);"ENDE LOGIC-ANALYSER" :END
```

```
440 GOTO390
450 FOR I = 49152 TO 49350
460 READ X : POKE I,X : S=S+X : NEXT
470 DATA 120,173, 17,208, 41,239,141, 17,208,162,  0,169
480 DATA 127,141, 13,221,169, 16, 44, 13,221,240,251,173
490 DATA   1,221,157,  0,193,232,208,240,173, 17,208,  9
500 DATA  16,141, 17,208,169,  1,160,  0,153,  0,216,153
510 DATA   0,217,153,  0,218,153,  0,219,200,208,241,169
520 DATA   0,133,247,169,193,133,248, 88, 76, 78,192,198
530 DATA 247, 76, 78,192,230,247,160, 36,162,  7,177,247
540 DATA 106, 72,176,  4,169,100,208,  2,169,102, 72,138
550 DATA  72, 10,170,189,182,192,133,249,189,183,192,133
560 DATA 250,104,170,104,145,249,104,202, 16,222,136, 16
570 DATA 215,160,  0,177,247,141,127,  7, 72,106,106,106
580 DATA 106, 41, 15, 24,105, 48,201, 58,144,  3, 24,233
590 DATA  56,141,120,  7,104, 72, 41, 15, 24,105, 48,201
600 DATA  58,144,  3, 24,233, 56,141,121,  7,104,160,  7
610 DATA 106, 72,169, 48,105,  0,153,160,  7,104,136, 16
620 DATA 243, 96,211,  6,131,  6, 51,  6,227,  5,147,  5
630 DATA  67,  5,243,  4,163,  4, 59
640 IF S <> 23552 THEN PRINT "FEHLER IN DATAS !!" : END
650 RETURN
```

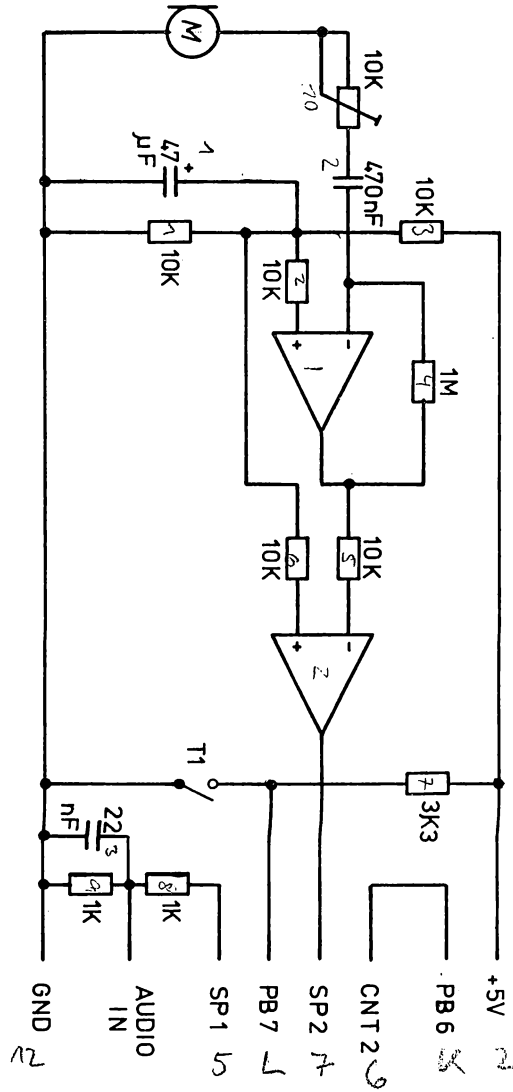
9.6 Der C64 lernt sprechen

Nachdem der C64 in seinem bisherigen Dasein von PAC-MAN über Textverarbeitung bis zum Meßgerät eine Menge gelernt hat, wollen wir ihm jetzt das Sprechen beibringen. Mit einer kleinen Zusatzschaltung und einem Mikrofon ist das nicht so schwer, wie bei einem Wellensittich. Denn unserem Computer müssen wir einen Text nur einmal vorsprechen, damit er ihn (fast) genau so wiederholt. Bei einem Wellensittich bleiben oft auch tage- und wochenlange Wiederholungsversuche fruchtlos. Womit auch gleich geklärt ist, wer von den beiden der Klügere ist.

Doch Spaß beiseite.
Über die Erzeugung von Sprache mit dem Computer ist in letzter Zeit häufig geschrieben worden. In den meisten Kapiteln zu diesem Thema wurde das Prinzip der Sprachgenerierung mit der sogenannten Phonemsynthese erläutert. Dieses Verfahren hat einigen sehr interessante Merkmale. Darum wollen wir kurz darauf zu Sprechen kommen.
Die Phonemsynthese geht von der Tatsache aus, daß sich ein gesprochenes Wort in kleine Laute unterteilen kann. Diese Laute bilden ähnlich den chemischen Elementen einen Grundstock, aus dem sich alle Stoffe oder, bei unserem Fall zu bleiben, alle Worte bilden lassen.
Allerdings gibt es mehr Laute oder eben Phoneme, als es Buchstaben gibt. Insgesamt reichen aber etwa 60 Phoneme um durchaus gut verständliche Sprache zu erzeugen.
Die Halbleiterindustrie liefert zum Teil sehr komfortabel zu handhabende Phonemgeneratoren, die alle benötigten Komponenten bereits auf einem Chip integriert haben. Die Auswahl eines Phonems benötigt dabei nur ein Byte. Dadurch ergibt sich natürlich ein sehr geringer Speicherbedarf pro gesprochenem Wort.

Leider sind diese ICs immer noch recht teuer. So liegt der Preis für das wohl zur Zeit am besten zu beschaffene IC, dem SC-01, bei etwa 100 - 120 DM.
Aber es gibt noch andere Methoden der Spracherzeugung mit zum Teil wesentlich geringeren Kosten. Unser Sprachgenerator kostet Sie vermutlich nur ein zehntel des zuvor genannten Preises.
Die von uns benutzte Methode ist zu vergleichen mit einem Diktiergerät. Zuerst werden die Worte in ein Mikrofon gesprochen und im Speicher des Rechners untergebracht. Anschließend kann dann der Computer die aufgezeicheten Texte nach belieben wiederholen.
Das vom Mikrofon gelieferte Sprachsignal wird zuerst verstärkt. Die Verstärkung wird sehr hoch gewählt. Dadurch werden alle Amplituden-, also Lautstärkeinformationen aus dem Sprachsignal entfernt und wir bekommen ein sauberes Rechtecksignal. Dies Rechecksignal können wir ohne Schwierigkeiten im Computer speichern. Geben wir das gespeicherte Signal anschließend wieder aus, wobei die Rechecksignale über ein Tiefpaß-Filter geführt werden, so erhalten wir ein gut verständliches Tonsignal mit großer Ähnlichkeit zum ursprünglichen Eingangssignal.

Soweit die Theorie unserer Spracherzeugung. Betrachten wir jetzt einmal, wie das Ganze in der Praxis passiert. Doch vorab eine kleine Bemerkung. Die folgende Bauanleitung und die abgedruckten Programme stellen nicht, wie beim Frequenzzähler oder Digitalvoltmeter eine vollständige und entgültige Bauanleitung dar. Vielmehr sollten Sie die Programme, aber auch die Schaltung nur als Anregung zu eigenem Experimentieren verstehen. Denn sicher kann durch eine verbesserte Schaltung oder durch Änderungen am Programm die Verständlichkeit deutlich verbessert werden.



Sprachein/ausgabe

Betrachten wir zunächst einmal die Schaltung.
Wieder werden OPs vom Typ LM 324 eingesetzt. Der erste OP verstärkt das vom Mikrofon kommende Tonsignal. Dabei kann mit dem 10K Poti die Empfindichkeit auf verschiedenen Mikrofone eingestellt werden. Die Einstellung erfolgt 'nach Gehör', d.h. auf beste Wiedergabe. Der zweite OP stellt einen Komparator dar. Mit diesem Komparator werden die verstärkten Mikrofonsignale begrenzt. Am Ausgang erhalten wir ein sauber begrenztes Rechtecksignal, wie es zur weiteren Verarbeitung mit dem C64 benötigt wird.
Der Taster T1 stellt eine 'Aufnahmetaste' dar. Um die Sprache aufzunehmen, muß dieser Taster gedrückt werden.
Die RC-Kombination stellt unser Tiefpaßfilter dar. Hier werden die steilen Flanken des vom C64 gelieferten Rechtecksignals etwas 'rundgebogen', die Sprache klingt nicht mehr so verzerrt. Den Kondensator mit dem Wert von 22 nF können Sie experimentell vergrößern oder verkleinern.
Das Ausgangssignal des Tiefpasses wird an den Eingang Audio-In an der Audio-Video-Buchse geführt. Dadurch haben wir die Möglichkeit der Tonbeeinflußung durch die im SID eingebauten Filter. Gerade diese Filterwerte beeinflußen die Verständlichkeit der Sprachausgabe enorm. Auch hier gilt: Versuch macht Klug.

Wie bekommen wir aber die Rechecksignale in den Rechner und wieder hinaus?
Nun, dafür gibt es grundsätzlich verschiedene Möglichkeiten. Zum einen können wir den Ausgang der Verstärkerstufe an ein User-Port-Bit legen. In einer kleinen Maschinensprache-Schleife können wir den User-Port regelmäßig abfragen und den eingelesenen Wert abspeichern. Das ist allerdings ein sehr unökonomischer Weg. Da unsere gewünschte Information in nur einem Bit enthalten ist, wäre es ausgesprochen verschwenderisch, wenn wir alle 8 Bits abspeichern würden.

Darum ist es besser, wenn in der Abfrageschleife 8 nacheinander gelesene Bits in einem Byte gespeichert werden können. Dadurch wird allerdings die Abfrageschleife komplizierter und länger. Gerade die Dauer der Schleife ist aber ein sehr entcheidendes Kriterium für die Qualität der anschließenden Wieergabe. Je kürzer die Abfrageschleife ist, um so öfter können wir ein Bit vom User-Port holen, das abgespeicherte Bitmuster ist der aufzuzeichnenden Sprache ähnlicher.

Warum aber umständlich programmieren, wenn der C64 uns mit seiner Hardware einen großen Teil der 'Dreckarbeit' abnehmen kann. Die in den CIAs integrierten seriellen Schieberegistern bieten sich für unseren Zweck ja geradezu an.

Bei diesen Schieberegistern wird im Eingabe-Modus abhängig vom Taktsignal am CNT-Eingang der SP-Eingang abgefragt. Nach 8 Taktsignalen ist das Schieberegister voll, ein Byte mit 8 Sprachbits kann aus dem Register 12 der CIA (SDR) abgeholt werden. Dabei wird im Register 13, dem Interrupt-control-Register (ICR) sogar durch Setzen des Bits 4 angezeigt, wenn die 8 Taktimpulse dagewesen sind. Unsere Software muß also nur im ICR nachsehen, ob das Bit 4 gesetzt ist. Ist dies der Fall, dann holen wir das Sprachbyte aus dem SDR und speichern es in unserem für die Sprache reservierten Speicherbereich.

Im Ausgabe-Modus ist der Verlauf ganz ähnlich. Jetzt wird ein Sprachbyte in das SDR geschrieben. Die Wandlung in das gewünschte serielle Format wird von der CIA übernommen. Das Bit 4 des ICR signalisiert nun, ob neue Daten in das Schieberegister geschrieben werden müssen. Das ist dann der Fall, wenn alle Bits hinausgeschoben wurden.

Nachdem nun geklärt ist, wie wir unsere Sprache in den C64 hinein und auch wieder heraus bekommen, stellt sich die Frage nach einem geigneten Speicherbereich. Da die Programmierung der CIA sowieso nur in Maschinensprache

schnell genug zu handhaben ist, können wir den ganzen RAM-Bereich unter dem Basic-Rom, dem Character-Rom und dem Kernal verwenden. Das sind immerhin 24K Ram, Platz für etwa 20 Sekunden Sprache und es geht Ihnen nicht ein Basic-Byte für den Sprachspeicher verloren!
Die verwendete Routine für Aufnahme und Wiedergabe ist genau 160 Bytes lang, paßt also schön in den Cassettenpuffer. Der Bereich $C000 bis $CFFF, in dem die meisten Maschinenroutinen dieses Buchs abgelegt sind, ist von uns natürlich nicht zu nutzen, da auch dieser Speicherbereich ja zur Sprachspeicherung benutzt wird.
Damit haben Sie weiterhin den gesamten Basic-Bereich zur Verfügung und können trotzdem die Sprachausgabe nutzen.

```
033C                    *=   $33C
0001          PROZPORT  =    1
00F7          POINTER   =    $F7
00F9          ENDE      =    $F9
DC00          CIA1      =    $DC00
DD00          CIA2      =    $DD00
DC04          TIMER1    =    CIA1 + 4
DD04          TIMER2    =    CIA2 + 4
DC0C          SDR1      =    CIA1 + 12
DD0C          SDR2      =    CIA2 + 12
DC0D          ICR1      =    CIA1 + 13
DD0D          ICR2      =    CIA2 + 13
DC0E          CRA1      =    CIA1 + 14
DD0E          CRA2      =    CIA2 + 14
DD01          PORT      =    CIA2 + 1
DD03          DR        =    CIA2 + 3
0032          TIME1     =    50
0064          TIME2     =    100
D020          RAHMEN    =    $D020
                                          ;ANFANG DER AUFNAHME-ROUTINE
033C 78       HEAR      SEI                ;BENUTZT CIA U2
033D A9 64              LDA  #<TIME2       ;
033F 8D 04 DD           STA  TIMER2        ;TIMER ARBEITET ALS TAKT
0342 A9 00              LDA  #>TIME2       ;FUER SDR
0344 8D 05 DD           STA  TIMER2+1
0347 AD 0D DD           LDA  ICR2          ;INTERRUPTCONTROL LOESCHEN
034A A9 13              LDA  #%0010011     ;BERIEBSART VON TIMER UND SDR
034C 8D 0E DD           STA  CRA2          ;FESTLEGEN
034F A0 00              LDY  #0
X 0351 A2 09            LDX  #$09          ;
0353 AD 01 DD TASTE     LDA  PORT          ;AUFNAHMETASTE GEDRUECKT?
0356 30 FB              BMI  TASTE
0358 C8       PRELL     INY                ;TASTE ENTPRELLEN
0359 D0 FD              BNE  PRELL
```

```
035B CA                 DEX
035C D0 FA              BNE  PRELL
035E AD 20 D0           LDA  RAHMEN       ;RAHMENFARBE ALS ZEICHEN
0361 85 FB              STA  $FB          ;FUER BEGINNENDE AUFNAHME
0363 A5 01              LDA  1            ;IN WEISS ÄNDERN
0365 8D 20 D0           STA  RAHMEN
                                          ;HIER START DER AUFNAHME
0368 AD 01 DD LOOP      LDA  PORT         ;TASTE GEDRUECKT?
036B 30 21              BMI  HEAREND      ;WENN NICHT, DANN ZUENDE
036D A9 08              LDA  #%1000       ;SDR-BIT IM ICR
036F 2C 0D DD SDRFULL   BIT  ICR2         ;EIN BYTE FERTIG?
0372 F0 FB              BEQ  SDRFULL
0374 AE 0C DD           LDX  SDR2         ;WERT AUS SDR LESEN
0377 A9 30              LDA  #$30         ;UMSCHALTEN AUF RAM
0379 85 01              STA  PROZPORT     ;IM BEREICH AB $A000
037B 8A                 TXA
037C 91 F7              STA  (POINTER),Y ;SDR-WERT INS RAM
037E A9 37              LDA  #$37         ;ZURUECKSCHALTEN AUF
0380 85 01              STA  PROZPORT     ;ROM
0382 E6 F7              INC  POINTER      ;
0384 D0 E2              BNE  LOOP         ;ADRESSZÄHLER ERHÖHEN
0386 E6 F8              INC  POINTER+1    ;BIS MAXIMAL $0000
0388 D0 DE              BNE  LOOP         ;
038A C6 F7              DEC  POINTER      ;WENN $0000 DANN
038C C6 F8              DEC  POINTER+1    ;ZURÜCK AUF $FFFF
038E A5 FB    HEAREND   LDA  $FB          ;ALTE RAHMENFARBE ZURÜCK
0390 8D 20 D0           STA  RAHMEN
0393 58                 CLI               ;UND ZURÜCK NACH BASIC
0394 60                 RTS
                                          ;BEGINN DER WIEDERGABE
0395 78       TALK      SEI               ;ÜBER CIA U1
0396 A9 32              LDA  #<TIME1      ;TIMER A ALS BAUD-RATE
0398 8D 04 DC           STA  TIMER1       ;GENERATOR
039B A9 00              LDA  #>TIME1
```

```
039D 8D 05 DC          STA  TIMER1 +1
03A0 A9 7F             LDA  #$7F         ;ICR-MASKE LÖSCHEN
03A2 8D 0D DC          STA  ICR1
03A5 AD 0D DC          LDA  ICR1         ;ICR LÖSCHEN
03A8 A9 51             LDA  #%01010001   ;BETRIEBSART TIMER & SDR
03AA 8D 0E DC          STA  CRA1
03AD A0 00             LDY  #0           ;SDR EINMAL ANSTOSSEN
03AF 8C 0C DC          STY  SDR1
03B2 A9 30    TALKLOOP LDA  #$30         ;AB $A000 AUF RAM UMSCHALTEN
03B4 85 01             STA  PROZPORT
03B6 B1 F7             LDA  (POINTER),Y ;BYTE AUS DEM SPEICHER HOLEN
03B8 AA                TAX
03B9 A9 37             LDA  #$37         ;UND ZURÜCKSCHALTEN
03BB 85 01             STA  PROZPORT
03BD A9 08             LDA  #%1000       ;SDR-BIT IM ICR
03BF 2C 0D DC SDREMPTY BIT  ICR1         ;SDR LEER?
03C2 F0 FB             BEQ  SDREMPTY
03C4 8E 0C DC          STX  SDR1         ;SPRACHBYTE INS SDR
03C7 E6 F7             INC  POINTER      ;ADRESSPOINTER ERHÖHEN
03C9 D0 02             BNE  CHKEND
03CB E6 F8             INC  POINTER+1
03CD A5 F7    CHKEND   LDA  POINTER      ;ENDE DER SPRACHAUSGABE
03CF C5 F9             CMP  ENDE         ;ERREICHT?
03D1 D0 DF             BNE  TALKLOOP
03D3 A5 F8             LDA  POINTER+1
03D5 C5 FA             CMP  ENDE+1
03D7 D0 D9             BNE  TALKLOOP
03D9 4C A3 FD          JMP  $FDA3        ;CIAS INITIALISIEREN FÜR
                                         ;TIMER-INTERRUPT
```

03D0

Um das Arbeiten mit diesem Maschinenspracheteil etwas zu erleichtern, finden Sie nachfolgend ein Basic-Programm, mit dem Sie bis zu 20 kleine Texte eingeben können. Jeder gesprochene Text erhält eine Nummer, unter der er später wieder abgehört werden kann. Nach der Aufnahme hat man zusätzlich die Möglichkeit, den gesprochenen Text sofort abzuhören. Sollte die Aufnahme nichts geworden sein, so kann sie wiederholt werden.
Wichtig für die Übergabe der Parameter für Start des RAMs und Ende des gesprochenen Textes sind die vier Zero-Pageadressen 247 bis 250. Die getroffene Zuweisung lautet folgendermaßen:

1. Aufnahme.

   In den Adressen 247 und 248 wird im Format Lowbyte/Highbyte die Startadresse für den Beginn der nächsten Aufzeichnung hinterlegt. Die kleinsten wählbaren Werte sind 0 für die Adresse 247 und 160 für 248 entsprechend der Adresse $A000 = dez. 40960. Nach der Aufzeichnung enthält diese Adresse die Adresse des letzten von der Aufzeichnung belegten Bytes im Sprachspeicher.
   Die Adressen 249 und 250 werden bei Aufnahme nicht verwendet.

2. Wiedergabe.

   In den Adressen 247 und 248 wird, wieder im Format Lowbyte/Highbyte, die Startadresse für das erste aus dem Sprachspeicher zu übertragende Byte angegeben. Die Adressen 249 und 250 enthalten die Adresse des letzten zu Übertragenden Sprachbytes. Das Format ist wie zuvor genannt.

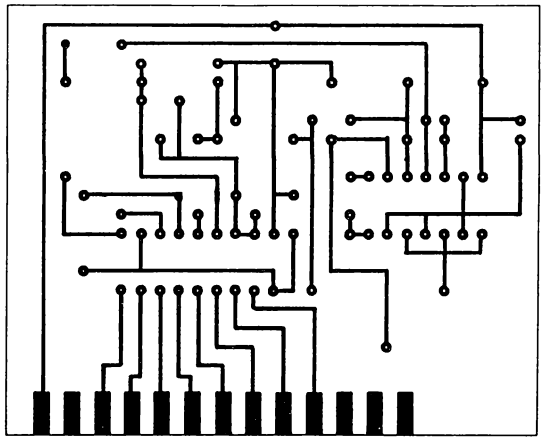
In den Zeilen 430 bis 450 des Basic-Programms wird der SID mit einigen Werten geladen. In der Zeile 430 wird die Filterfrequenz der eingebauten Filter bestimmt. In der Zeile 440 wird mit dem Wert 8 das am Eingang Ext.In anliegende NF-Signal über die Filter beeinflußt. Der zweite Wert, im Programm mit 240 angegeben, bestimmt die Höhe der Resonanz bei der angegebenen Filterfrequenz. In der Zeile 450 wird zum Schluß die Lautstärke (15) und die Filterart, also Hochpaß, Tiefpaß, Bandpaß oder eine Kombination der drei bestimmt.
Mit diesen Werten sollten Sie unbedingt experimentieren, da damit sehr weitreichende Beeinflussungen des Klangbildes und der Verständlichkeit vorgenommen werden können.

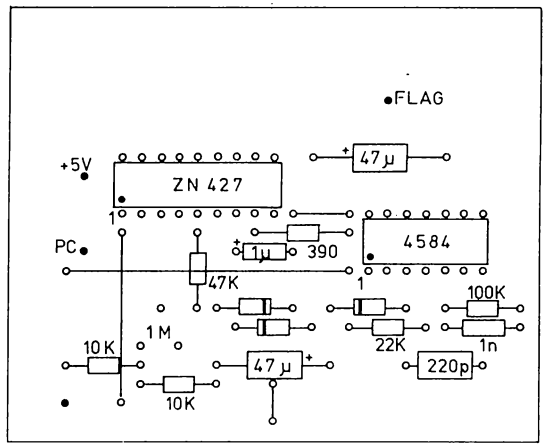
Dieses Basic-Programm bietet nur die wichtigsten Möglichkeiten, die für eine Erprobung der Schaltung und für Experimentierzwecke erforderlich sind. Ein gößeres oder umfangreicheres Programm ist aber auch nicht sinnvoll, da die Sprachausgabe an sich ja keinen Selbstzweck darstellt. Sinnvoll wird sie erst im Zusammenhang mit anderen Aufgaben und Programmen. Hier seinen nur beispielhaft die Programmierung einer sprechenden Uhr, der Einsatz in Spielen oder auch der Einsatz in Anwenderprogrammen erwähnt.
Wie wäre es z.B. mit einem Anwenderprogramm, sagen wir einer Lagerverwaltung, bei der der Computer den Benutzer höflich aber bestimmt durch ein deutlich vernehmbares "EINGABEFEHLER BITTE KORRIGIEREN" oder "JETZT DATENDISKETTE EINLEGEN" zu bestimmten Aufgaben auffordert.

```
100 DIM S(20)
110 S(1) = 40960
120 I = 1
130 SID = 54272  :GOSUB 510
140 PRINTCHR$(147) : PRINT : PRINT
150 PRINT "      F1  AUFNAHME  "
160 PRINT "      F3  WIEDERGABE"
170 PRINT "      F7  ENDE      "
180 GET A$ : IFA$ ="" THEN 180
190 IF A$ = CHR$(133) THEN GOTO 230
200 IF A$ = CHR$(134) THEN GOTO 350
210 IF A$ = CHR$(136) THEN GOTO 490
220 GOTO 180
230 PRINT CHR$(147)
240 H = S(I)/256   :POKE 248,H
250 L = S(I)-H*256 :POKE 247,L
260 PRINT "START TEXT  #";I;"IST ADRESSE";S(I)
270 SYS 828
280 H = PEEK(248)
290 L = PEEK(247)
300 S (I+1) = (H*256)+L
310 PRINT "GESPEICHERT #";I;"BIS ADRESSE";S(I+1)
320 INPUT "GLEICH ABHOEREN ";A$
330 IF A$<>"J" THEN I=I+1 : GOTO 140
340 N = I:GA = 1:GOTO 370
350 PRINT CHR$(147)
360 INPUT "BITTE TEXT# EINGEBEN";N
370 IFS(N+1)<40960THENPRINT"UNTER DIESER # KEIN TEXT":ER=1
380 IF ER=1 THEN ER=0:INPUT "ENTER FUER WEITER";A$:GOTO350
390 H = S(N)/256     :POKE 248,H
400 L = S(N)-H*256   :POKE 247,L
410 H = S(N+1)/256   :POKE 250,H
420 L = S(N+1)-H*256 :POKE 249,L
430 POKE SID+22,150
```

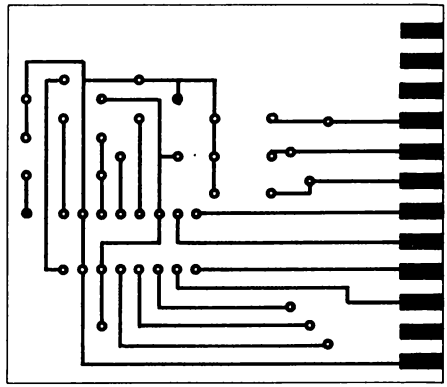
```
440 POKE SID+23,8+240
450 POKE SID+24,15+32
460 SYS 917
470 IF GA=1 THEN GA=0:INPUT"TEXT OK";A$:IFA$="J"THENI=I+1
480 GOTO 140
490 PRINT CHR$(147)"ENDE DES PROGRAMMS"
500 END
510 FOR I = 828 TO 987
520 READ X : POKE I,X : S=S+X : NEXT
530 DATA 120,169,100,141,  4,221,169,  0,141,  5,221,173
540 DATA  13,221,169, 19,141, 14,221,160,  0,162, 16,173
550 DATA   1,221, 48,251,200,208,253,202,208,250,173, 32
560 DATA 208,133,251,165,  1,141, 32,208,173,  1,221, 48
570 DATA  33,169,  8, 44, 13,221,240,251,174, 12,221,169
580 DATA  48,133,  1,138,145,247,169, 55,133,  1,230,247
590 DATA 208,226,230,248,208,222,198,247,198,248,165,251
600 DATA 141, 32,208, 88, 96,120,169, 50,141,  4,220,169
610 DATA   0,141,  5,220,169,127,141, 13,220,173, 13,220
620 DATA 169, 81,141, 14,220,160,  0,140, 12,220,169, 48
630 DATA 133,  1,177,247,170,169, 55,133,  1,169,  8, 44
640 DATA  13,220,240,251,142, 12,220,230,247,208,  2,230
650 DATA 248,165,247,197,249,208,223,165,248,197,250,208
660 DATA 217, 76,163,253
670 IF S <> 23043 THEN PRINT "FEHLER IN DATAS !!" : END
680 RETURN
```

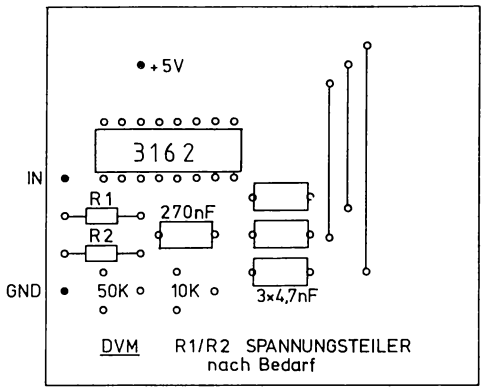
Zum Abschluß dieses Buches folgen noch einige Leiterplatten-Layouts. Wer von den Lesern die handwerklichen Möglichkeiten besitzt, kann die abgedruckten Lagepläne direkt als 1:1 Vorlage verwenden.
Allerdings sei an dieser Stelle ausdrücklich erwähnt, daß der Aufbau auf Lochrasterplatten mit der Fädel- oder der Wire-Wrap-Technik im allgemeinen wesentlich platzsparender und schneller ist. Aus diesem Grunde wurde auch auf einen Entwurf einer Leiterplatte für den Prommer verzichtet.
Ebenso wie bei der Epromkarte wäre beim Prommer schon aus Anschlußgründen eine doppelseitige Leiterplatte nötig gewesen. Das exakte Reproduzieren beidseitiger Leiterplatten übersteigt aber vermutlich die Möglichkeiten der meisten Hobbyelektroniker.

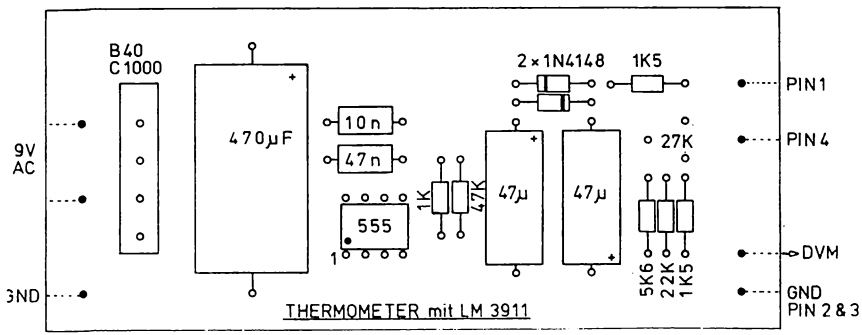


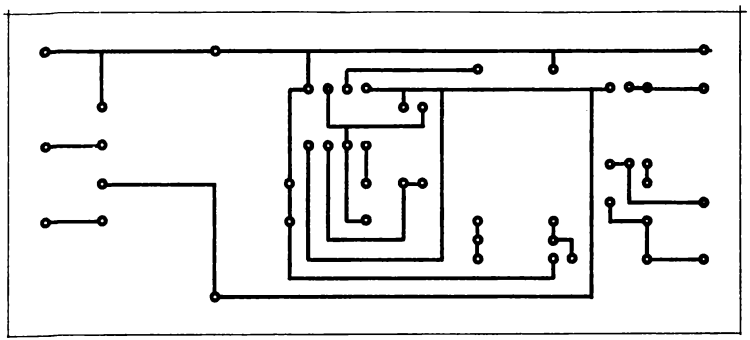

FLAG
+5V
ZN 427
47µ
1
PC
1µ
390
4584
47K
1
100K
1 M
10 K
22K
1n
47 µ
220p
10K

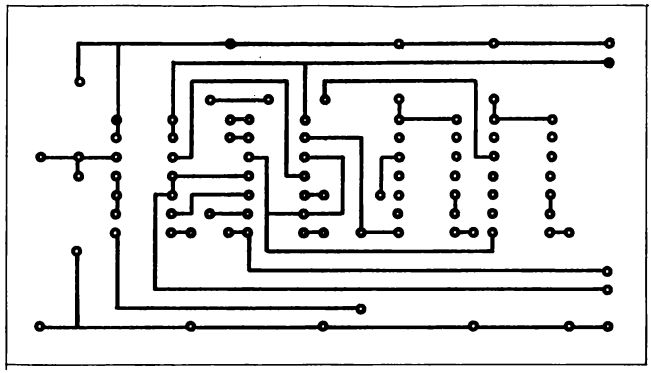
IN

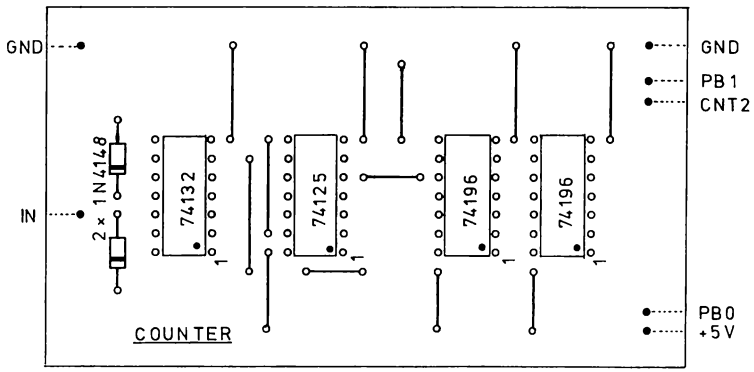


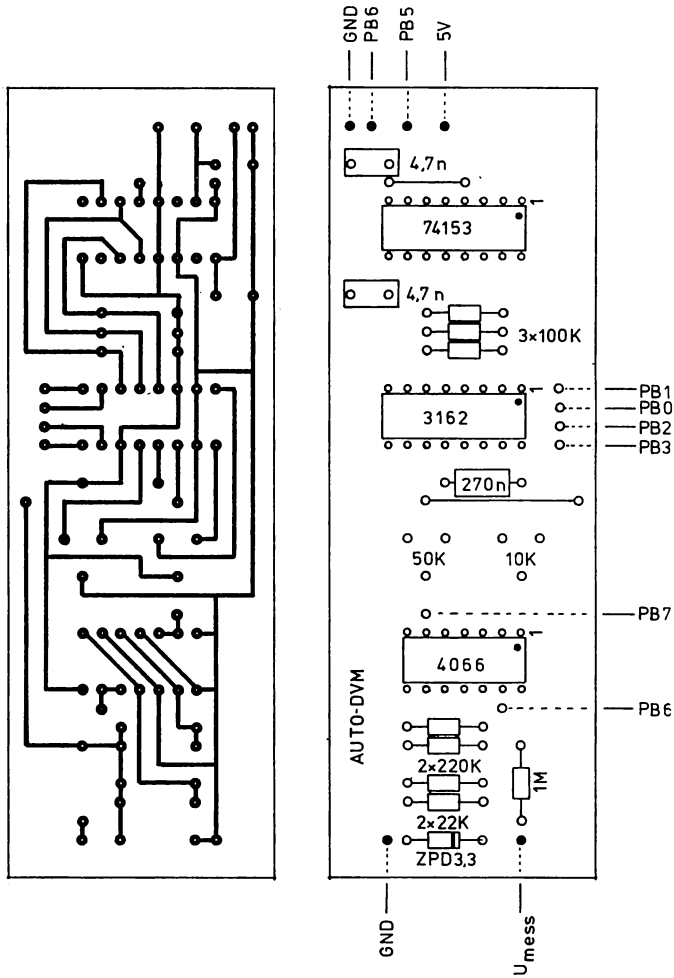

+5V
3162
IN
R1
270nF
R2
GND
50K
10K
3x4,7nF
DVM
R1/R2 SPANNUNGSTEILER
nach Bedarf


B40
C1000
470µF
10n
47n
555
1K
47K
47µ
47µ
2×1N4148
1K5
27K
5K6
22K
1K5
9V
AC
GND
PIN1
PIN4
DVM
GND
PIN 2&3
THERMOMETER mit LM 3911






GND
IN
2 × 1N4148
74132
74125
74196
74196
GND
PB1
CNT2
PB0
+5V
COUNTER


GND
PB6
PB5
5V
4,7n
74153
4,7 n
3x100K
3162
PB1
PB0
PB2
PB3
270n
50K
10K
PB7
4066
PB6
AUTO-DVM
2x220K
1M
2x22K
ZPD3,3
GND
Umess

FANTASTISCH

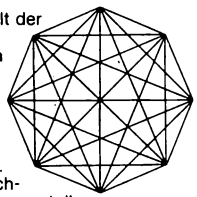
Was so ein COMMODORE 64 mit DATA BECKER PROGRAMMEN alles kann:

Mit ***DATAMAT*** „frißt" Ihr C-64 Ordner, Karteikästen und Notizbücher. DATAMAT ist eine universelle Dateiverwaltung, die Sie auf vielfältige Weise nutzen können. Frei gestaltbare Eingabemaske mit bis zu 50 Feldern, max. 40 Zeichen pro Feld und bis zu 253 Zeichen pro Datensatz. Bis zu 2000 Datensätze pro Diskette. Sortiermöglichkeit nach mehreren Feldern in beliebiger Kombination. Druck von Auswertungen, Listen und Etiketten. DATAMAT sollte zu jedem 64er gehören.

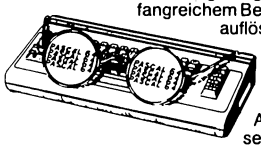
Mit ***TEXTOMAT*** werden Briefe, Rundschreiben und komplette Bücher zum Kinderspiel. TEXTOMAT schafft 80 Zeichen pro Zeile durch horizontales Scrolling, Ausdruck bis 255 Zeichen Breite, Textlänge bis zu 24 000 Zeichen im Speicher, Verketten von Texten, Textbausteinverarbeitung, Formatierung, Blocksatz, Formularsteuerung, Serienbriefe und natürlich deutsche Zeichen nicht nur auf dem Bildschirm, sondern mit vielen Druckern (Epson, GP 100 VC, 1525, 1526, 801) auch auf dem Papier. Mit TEXTOMAT macht Schreiben Spaß.

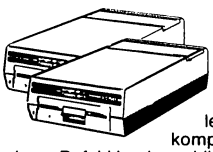
SYNTHIMAT verwandelt Ihren COMMODORE 64 in einen professionellen, polyphonen, dreistimmigen Synthesizer, der in seinen unglaublich vielen Möglichkeiten großen und teuren Synthesizern kaum nachsteht. Mit SYNTHIMAT wird Ihr 64 für wenig Geld zur Supermaschine.

Entdecken Sie die faszinierende Welt der Computergraphik mit ***SUPERGRAPHIK 64,*** der starken Befehlserweiterung mit den vielseitigen Möglichkeiten. 187 (!) Befehlskombinationen für Sprites, Graphik und Sound. Mit der SUPERGRAPHIK 64 machen Sie mehr aus Ihrem COMMODORE 64. Für Druckerbesitzer gibt es die Möglichkeit, eine Hardcopy des Bildschirms zu erstellen.



PASCAL 64 ist ein leistungsfähiger PASCAL-Compiler, mit umfangreichem Befehlssatz, der auch die hochauflösende Graphik und die Sprites des COMMODORE 64 unterstützt. Ein-/Ausgabe über Diskette und Drucker sowie REAL und INTEGER Arithmetik. PASCAL 64 ist sehr schnell, da echter Maschinencode erzeugt wird!

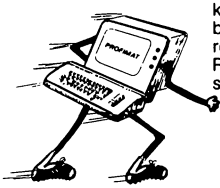




DISKOMAT hilft Ihnen, mehr aus Ihrer Floppy zu machen, mit SUPERTWIN, dem Steuerprogramm, das zwei VC-1541 wie ein Doppellaufwerk verwaltet, mit DISC-BASIC, den Diskettenbefehlen des BASIC 4.0, mit denen Sie eine komplette Diskette oder Auszüge mit einem Befehl kopieren können und mit einem komfortablen DISK-MONITOR.

Mit ***FAKTUMAT*** ist das Schreiben von Rechnungen kein Alptraum mehr. Eine Sofortfakturierung mit integrierter Lagerbuchführung. Individuelle Anpassung von Steuersätzen, Maßeinheiten und Firmendaten. Kunden- und Artikelstamm voll pflegbar. Schneller Zugriff auf Kunden- und Artikeldaten über frei definierbaren, 6-stelligen Schlüssel. Automatische Fortschreibung von Artikel- und Kundendaten, individuell nutzbar. Alles in allem die Arbeits- und Zeitersparnis die Sie sich schon längst gewünscht haben.

Mit Maschinensprache geht vieles schneller. ***PROFIMAT*** enthält den komfortablen Maschinensprache Monitor PROFI-Mon und PROFI-Ass, einen sehr leistungsfähigen Assembler. PROFI-Ass bietet unter anderem formatfreie Eingabe, komplette Assemblerlistings, ladbare Symboltabellen (Labels), redifinierbare Symbole, eine Reihe von Assembleranweisungen, bedingte Assemblierung und Assemblerschleifen.



Darauf haben Sie bestimmt gewartet: Passend zum Superhit COMMODORE 64 gibt's jetzt die neuen DATA BECKER PROGRAMME - Spitzensoftware auf Diskette zu unglaublich niedrigen Preisen. Alle Programme werden mit ausführlichem Handbuch und in einem formschönen Ordner geliefert.

PROFIMAT IN STICHWORTEN

PROFIMAT ist ein leistungsfähiges Programmpaket zur professionellen Maschinenprogrammierung mit dem COMMODORE 64

- PROFI-MAT
- PROFI-MON
- 2 K-Byte Maschinenprogramm
- Register- und Speicherinhalte anzeigen/ändern
- Maschinenprogramme laden, speichern, ausführen, disassemblieren
- Speicherbereiche durchsuchen, vergleichen, füllen
- PROFI-ASS
- 4 K-Byte Maschinenprogramm
- formatfreie Eingabe
- komplette Assemblerlistings
- ladbare Symboltabellen
- bedingte Assemblierung und Assemblerschleifen
- Verkettung von Quellprogrammen
- umfangreiche Pseudo-Opcodes
- vielfältige arithmetische und logische Operationen
- arbeitet mit beliebiger Gerätekonfiguration
- Diskettenprogramm
- mit ausführlichem Handbuch

PASCAL 64 IN STICHWORTEN

PASCAL 64, DAS IDEALE PASCAL FÜR EINSTEIGER ZUM COMMODORE 64

- erzeugt echte Maschinensprache
- ca. 22 K frei für Programm und Symboltabelle
- einfache Programmerstellung durch Bildschirmeditor
- enthält alle wichtigen Methoden zur Programmstrukturierung

Standard PASCAL-Features:
- BEGINN . . . END
- WHILE . . . DO
- REPEAT . . . UNTIL
- CASE
- PROCEDURE
- FUNKTION

Datentypen:
- CHAR
- INTEGER
- REAL
- ARRAY

Sonderfunktionen:
- PEEK
- SYS
- Graphikbefehle
- Behandlung sequentieller Dateien
- einbinden externer Prozeduren und Funktionen von Diskette
- lieferbar auf Diskette
- mit detailliertem Handbuch

SYNTHIMAT IN STICHWORTEN

SYNTHIMAT ist ein leistungsfähiger, polyphoner Synthesizer für den COMMODORE 64

- drei Oszillatoren (VCOs) mit 7 Fußlagen und 8 Wellenformen
- drei Hüllkurvengeneratoren (ADSRs)
- ein Filter (VCF) mit 8 Betriebsarten und Resonanzregulierung
- VCF mit Eingang für externe Signalquelle
- ein Verstärker (VCA)
- Ringmodulation mit allen drei VCOs
- 8 softwaremäßig realisierte Oszillatoren (LFOs)
- kräftiger Klang durch polyphones Spielen
- zwei Manuale (Solo und Begleitung)
- speichern von bis zu 256 Klangregistern
- schneller Registerwechsel
- speichern von 9 Registerdateien Diskette
- „Bandaufnahme" auf Diskette durch direktes Spielen
- keine lästige Noteneingabe
- speichern von bis zu 9 „Bandaufnahmen" je Diskette
- integrierte 24 Stunden-Echtzeituhr
- einstellbares PITCH-BENDING
- farblich gekennzeichnete, übersichtlich angeordnete Module
- umfangreiches Handbuch
- läuft mit einem Diskettenlaufwerk
- Diskettenprogramm

DISKOMAT IN STICHWORTEN

DISKOMAT, ein Programmpaket mit SUPERTWIN, DISK-MONITOR und DISK-BASIC zum COMMODORE 64.

- SUPERTWIN, ein Steuerprogramm, mit dem Sie zwei VC 1541 Diskettenlaufwerke wie ein Doppellaufwerk benutzen. Kopieren von Dateien und zwischen zwei Laufwerken Backup-Möglichkeit. Kompatibel zu Programmen, die für Doppellaufwerke gedacht sind.
- DISK-BASIC bietet Ihnen die komfortablen Diskettenbefehle des BASIC 4.0.

APPEND	BACKUP	CATALOG	COLLECT
CONCAT	COPY	DOPEN	DCLOSE
DLOAD	DSAVE	HEADER	RECORD
SCRATCH	DIRECTORY	RENAME	DS & DS $

- DISK-MONITOR
- Lesen eines Blocks von Diskette
- Anzeige und Ändern eines Blocks am Bildschirm
- Schreiben eines Blocks auf Diskette
- Senden von Diskettenbefehlen
- Anzeigen der Fehlermeldung der Diskette
- Diskettenprogramm mit ausführlichem deutschen Handbuch

SUPERGRAPHIK IN STICHWORTEN

SUPERGRAPHIK 64 ist ein umfassendes Graphikprogramm für den COMMODORE 64

— 2 unabhängige, hochauflösende Graphik-Seiten (320 x 200 Punkte)
— 1 Standard Low-Graphik-Seite (80 x 50 Punkte)
— Normalfarben-Graphik (320 x 200 Punkte)
— Multicolor-Graphik (160 x 200 Punkte)
— verdecktes Zeichnen (z. B. Text sichtbar, Graphikseite 2 wird erstellt)
— 183 Befehle und Befehlskombinationen:

1. Für jeden Befehl wählbare Zwischenmodi: Zeichnen, Löschen, Punktieren, Graphik-Cursor bewegen, Zeichnen mit/ohne Farbsetzung, Punkte zählen
2. Durch einfache Befehle zu steuernde Graphikfiguren: Punkt, Linie, Linienschar, Linie vom Graphik-Cursor, Kreise, Kreisbögen, Ellipse, Ellipsenbögen, selbstdefinierbare Figuren, rotieren und vergrößern dieser Figuren, Rahmen, Feld, Text in Graphik
3. Weitere Graphik-Befehle: Graphikseiten- und Moduswechsel, Graphik löschen, Graphik invertieren, Scrolling von Text und Graphik, Wählen der Rahmen-, Hintergrund-, Zeichen- oder Punktfarbe

— Speichern, Laden von Graphik (auch verdeckt)
— Hardcopies für EPSON, CBM 1526, Seikosha GP 100 VC, Farb(!)drucker Seikosha GP 700 und andere mit DATA BECKER Interface
— 8 Sprites definiert durch einen Befehl!
— alle Sprite-Eigenschaften veränderbar
— Positionieren und Bewegen (!) von 8 Sprites gleichzeitig und unabhängig voneinander, während das übrige Programm weiterläuft (IRQ)
— Sprite-Kollisionsüberprüfung, Joystickunterstützung
— automatische Unterbrechung des BASIC-Programms bei Kollisionen (Interrupt), Sprung in Unterbrechungsroutine, dann Weiterführung des Hauptprogramms
— komfortable Soundprogrammierung mit Verstellung aller möglichen Sound-Parameter (Laufstärke, Klang, Filter, Tonhöhe, Tonlänge) ebenfalls unabhängig vom übrigen Programmablauf
— umfangreiche Anleitung
— Diskettenprogramm

DATAMAT IN STICHWORTEN

DATAMAT ist ein komfortables Dateiverwaltungsprogramm für den COMMODORE 64

— menuegesteuert, dadurch extrem einfach zu bedienen
— für jede Art von Daten zu gebrauchen
— völlig frei gestaltbare Eingabemaske
— 50 Felder pro Datensatz
— 253 Zeichen pro Datensatz
— je nach Umfang der Datensätze bis zu 2000 pro Datei
— Schnittstelle zu TEXTOMAT
— lieferbar als Diskettenprogramm mit ausführlichem deutschen Handbuch
— läuft mit ein oder zwei Floppies

Sie können:

— nach beliebigen Feldern selektieren
— nach allen Feldern gleichzeitig sortieren
— Listen in völlig freiem Format drucken
— Etiketten drucken
— mit jedem COMMODORE Drucker arbeiten
— nahezu jeden Fremddrucker verwenden (mit DATA BECKER Interface)

TEXTOMAT IN STICHWORTEN

TEXTOMAT, ein außergewöhnliches Textverarbeitungsprogramm für den COMMODORE 64

— Diskettenprogramm
— mit umfangreichem deutschen Handbuch
— durchgehend menuegesteuert
— deutscher Zeichensatz auch auf Commodore-Druckern
— Rechenfunktionen
— 24000 Zeichen pro Text
— beliebig lange Texte durch Verknüpfung
— frei programmierbare Steuerzeichen
— horizontales Scrolling für 80 Zeichen pro Zeile
— läuft mit ein oder zwei Floppies
— Formularsteuerung für Randeinstellung usw.
— komplette Bausteinverarbeitung
— Blockoperationen, Suchen und Ersetzen
— Serienbriefschreibung mit DATAMAT
— formatierte Ausgabe auf Bildschirm
— an fast jeden Drucker anpassbar

STRUKTO 64 IN STICHWORTEN

STRUKTO 64 ist eine fantastische neue Programmiersprache für strukturiertes Programmieren mit COMMODORE 64.

— Interpretersprache, die die Vorzüge von BASIC UND PASCAL vereint
— strukturiertes Programmieren
— übersichtliche Programme
— leichte Erlernbarkeit
— einfache Bedienung
— eingebautes Toolkit erleichtert das Eingeben und Verbessern von Programmen
— leichteres Arbeiten mit der Floppy
— Sprite-Editor ermöglicht das Einlesen der Sprite-Formen direkt vom Bildschirm
— Graphikbedienung wird mit gut durchdachten Befehlen unterstützt
— Abspielen von Musik ist unabhängig vom Programmablauf möglich
— ca. 80 neue Befehle
— lieferbar als Diskettenprogramm
— ausführliches, deutsches Handbuch

STRUKTO 64 ist für alle Programmierer geeignet, die den COMMODORE 64 als Allround-Computer einsetzen wollen und auf einfache Weise anspruchsvolle Programme erstellen wollen.

KONTOMAT IN STICHWORTEN

KONTOMAT ein Diskettenprogramm zur Einnahme-/Überschußrechnung

- professionelles Programm, angelehnt an eine echte Finanzbuchhaltung
- maximal 120 definierbare Konten
- 4 Mehrwert- und Vorsteuerkennzeichen
- intervallmäßige Belegeingabe
- 4 verschiedene Buchungsarten (SOLL/HABEN, HABEN/SOLL, SOLL, HABEN)
- Anzeige der SOLL- und HABEN-Summe bei mehrfachen Buchungssätzen
- komfortable Belegeingabe mit Datum, Belegnummer, Buchungstext, Steuerkennzeichen, Buchungsart und Betrag
- Druck des Journals mit der Belegeingabe
- Druck von Kontenblättern
- Druck einer Summen- und Saldenliste mit Monats- und Jahresumsatzsummen
- betriebswirtschaftliche Auswertung mit Druckausgabe
- Verzeichnis der AfA und der Sofortabschreibung
- ein oder zwei Laufwerke
- umfangreiches Handbuch

FAKTUMAT IN STICHWORTEN

FAKTUMAT ist ein Programm zur Fakturierung und Rechnungserstellung für den COMMODORE 64

- voll menuegesteuert
- Artikeldatei & Kundendatei parameterisiert
- läuft auf 1 oder 2 Floppies 1541
- Drucker: VC-1525, VC-1526, EPSON RX-/FX-80 mit DATA BECKER Interface
- Diskettenprogramm
- frei definierbarer Firmenkopf
- 4 wählbare Mehrwertsteuersätze
- Maximal 950 Artikel bei 50 Kunden oder 50 Artikel bei 950 Kunden. Die Summe von Kunden + Artikel kann maximal 1000 betragen
- Druck von Rechnung und Lieferschein möglich
- Lagerbuchführung ist integriert. Alle Artikel die über Rechnung/Lieferschein rausgegangen sind, werden in der Lagerdatei abgebucht.
- Rechnungsbeträge werden in der Kundendatei festgehalten
- deutsches, detailliertes Handbuch
- Sie arbeiten mit einer Diskette für Ihre Daten

PAINT-PIC IN STICHWORTEN

PAINT-PIC ist ein faszinierendes Malprogramm für den COMMODORE 64

- Programmsteuerung: Tastatur
- Steuerung des Stifts: Cursortasten und eckige Klammer (diag.) (Joystick kann benutzt werden)
- Routinen: Linien, Rechtecke, Dreiecke, Parallelogramme, Kreise, Kreisbögen, Ellipsen, Bestimmung v. Mittelpunkt, u. perspektivischer Linie, Kopieren u. drehen v. Teilbildern, Verdoppeln, halbieren u. spiegeln v. Teilbildern
- Modi: Malstiftmodus (schmale Linie) Pinselmodus (8 verschiedene Breiten) (Art der Linie selbst definierbar)
- Textmodus (Kompl. Zeichensatz Commodore) (Hoch-Tiefschrift)
- Speichern: Teilbilder (Blöcke) oder ganze Bilder
- Menue: 1 Hauptmenue mit 8 Untermenues
- mit ausführlichem deutschen Handbuch
- Diskettenprogramm
- Bilder kann man auf Diskette oder Cassette abspeichern

MASTER 64 IN STICHWORTEN

MASTER: Professionelles Programmentwicklungssystem für COMMODORE 64

- 70 zusätzliche Befehle
- Bildschirmmaskengenerator
- Definieren von Bildschirmzonen
- Eingabe aus Zonen
- Formatierte Ausgabe
- Abspeicherung von Bildschirminhalten
- Arbeiten mit mehreren Bildschirmmasken
- ISAM Dateiverwaltung
- Datensätze bis zu 254 Zeichen
- Schlüssellänge bis zu 30 Zeichen
- Dateigröße nur von Diskettenkapazität abhängig
- Zugriff über Schlüssel und Auswahlmasken
- Druckmaskengenerator
- Erstellung beliebiger Formulare und Ausgabemasken
- BASIC-Erweiterungen
- Toolkitfunktionen
- Mehrfachgenaue Arithmetik Rechnen mit 22 Stellen Genauigkeit

Alles, was Sie schon immer über Ihren COMMODORE wissen wollten!

DAS
MASCHINENSPRACHE-
BUCH ZUM
COMMODORE 64
1984, ca. 200 Seiten,
DM 39,-

Eine leicht verständliche Einführung in die Programmierung in Maschinensprache für alle, denen die Programmierung des Commodore 64 in BASIC nicht mehr ausreicht. Sie lernen Aufbau, Arbeitsweise und Register des 6510-Mikroprozessors kennen und werden über Bit, Bytes, Daten und Adressen aufgeklärt. Anhand von Beispielen lernen Sie alle Befehle des 6510 kennen und anwenden. Dabei werden die Analogien zu BASIC Ihnen beim Verständnis helfen. Ein weiteres Kapitel beschäftigt sich mit der Eingabe von Maschinenprogrammen. Dort erfahren Sie auch alles über Monitor-Programme sowie über Assembler. Zum einfachen und komfortablen Erstellen Ihrer eigenen Maschinensprache enthält das Buch einen kompletten Assembler, damit Sie gleich von Anfang an komfortabel und effektiv programmieren können. Weiterhin finden Sie dort einen Disassembler, mit dem Sie sich Ihre Maschinenprogramme oder die Routinen des BASIC-Interpreters und des Betriebssystems ansehen können. Ein besonderer Clou ist ein in BASIC geschriebener Einzelschrittsimulator, mit dem Sie Ihre Programme schrittweise ausführen können. Dabei werden Sie nach jedem Schritt über Registerinhalte und Flags informiert und können den logischen Ablauf Ihres Programms verfolgen. Eine unschätzbare Hilfe besonders für den Anfänger. Als Beispielprogramm finden Sie ausführlich beschriebene Routinen zur Grafikprogrammierung und für BASIC-Erweiterungen. Natürlich sind alle Beispiele und Programme auf Ihren Commodore 64 zugeschnitten.

DAS TRAININGSBUCH
ZUM SIMON's BASIC,
1984, ca. 300 Seiten,
DM 49,-

SIMON's BASIC ist ein Hit — wenn man es richtig nutzen kann. Deshalb gibt es jetzt zu dieser vielseitigen Befehlserweiterung unser umfangreiches Trainingsbuch. Auf über 300 Seiten erklärt es Ihnen detailliert den Umgang mit den über 100 Befehlen des SIMON's BASIC. Alle Befehle werden ausführlich dargestellt, auch die, die nicht im Handbuch stehen! Natürlich zeigen wir auch die Masken des SIMON's BASIC und geben wichtige Hinweise, wie man diese umgeht. Zum Buch gehören zahlreiche Beispielprogramme und viele interessante Programmiertricks. Alle Befehle sind entsprechend ihren Anwendungen in entsprechende Kapitel geordnet: Einführung in das CBM BASIC 2.0 — Programmierhilfen — Fehlerbehandlung — Programmschutz — Programmstruktur — Variablen — Zahlenbehandlung — Eingabekontrolle — Ein/Ausgabe Peripheriebefehle — Graphik — Zeichensatzerstellung — Sprites — Musik — Steuernde Peripherie — SIMON's BASIC und die Verträglichkeit mit anderen Erweiterungen und Programmen. Dazu ein umfangreicher Anhang. Nach jedem Kapitel finden Sie Testaufgaben zum optimalen Selbststudium und zur Lernerfolgskontrolle. DAS TRAININGSBUCH ZUM SIMON's BASIC sollte jeder Anwender dieser universellen Befehlserweiterung unbedingt haben.

DATA BECKER BÜCHER

DAS GROSSE
FLOPPY-BUCH,
1983, ca. 320 Seiten,
DM 49,-

Darauf haben Sie gewartet: Endlich ein Buch, das Ihnen ausführlich und verständlich die Arbeit mit der Floppy VC-1541 erklärt. DAS GROSSE FLOPPY BUCH ist für Anfänger, Fortgeschrittene und Profis gleichermaßen interessant. Sein Inhalt reicht von der Programmspeicherung bis zum DOS-Zugriff, von der sequentiellen Datenspeicherung bis zum Direktzugriff, von der technischen Beschreibung bis zum ausführlich dokumentierten DOS-Listing, von den Systembefehlen bis zur detaillierten Beschreibung der Programme der Test/Demodiskette. Exakt beschriebene Beispiel- und Hilfsprogramme ergänzen dieses neue Superbuch. Aus dem Inhalt: Der erste Kontakt — Das Speichern von Programmen — Die Floppy-Systembefehle — Sequentielle Datenspeicherung — Relative Datenspeicherung — Die Fehlermeldungen und Ihre Ursachen — Der Direktzugriff — Der Zugriff auf das DOS — Technik der Floppy und der Diskette — DOS-Listing der VC-1541 — Dienstprogramme — BASIC-Erweiterungen und Programme zur komfortablen Benutzung der VC-1541 - Overlaytechnik — Diskmonitor — IEC-Bus und serieller Bus — Gemeinsamkeiten mit den großen CBM-Floppies und Unterschiede gegenüber der VC-1541. Mit dem GROSSEN FLOPPY-BUCH meistern Sie auch Ihre Floppy.

64 FÜR PROFIS
1983, ca. 280 Seiten,
DM 49,-

Wer besser und leichter in BASIC programmieren möchte, der braucht dieses Buch. 64 FÜR PROFIS zeigt, wie man erfolgreich Anwendungsprobleme in BASIC löst und verrrät die Erfolgsgeheimnisse der Programmierprofis. Vom Programmentwurf über Menüsteuerung, Maskenaufbau, Parameterisierung, Datenzugriff und Druckausgabe bis hin zur guten Dokumentation wird anschaulich mit Beispielen dargestellt, wie Profiprogrammierung vor sich geht. Besonders stolz sind wir auf die völlig neuartige Datenzugriffsmethode QUISAM, die in diesem Buch zum ersten Mal vorgestellt wird. QUISAM erlaubt eine beliebige Datenzusatzlänge, die dynamisch mit der Eingabe der Daten wächst. Eine lauffertige Literaturstellenverwaltung veranschaulicht die Arbeitsweise von QUISAM. Neben diesem Programm finden Sie auch noch weitere Programme zur Lager- und Adressenverwaltung, Textverarbeitung und einen Reportgenerator. Alle diese Programme sind mit Variablenliste versehen und ausführlich beschrieben. Damit sind diese für Ihre Erweiterungen offen und können von Ihnen an Ihre speziellen Bedürfnisse angepaßt werden. Mit 64 FÜR PROFIS steigen Sie in die Welt der Programmierprofis ein.

MIT DATA BECKER BÜCHERN MACHEN

SIE MEHR AUS IHREM COMMODORE

DATA BECKER BÜCHER

64 TIPS & TRICKS,
2. Auflage,
1983, ca. 290 Seiten,
DM 49,-

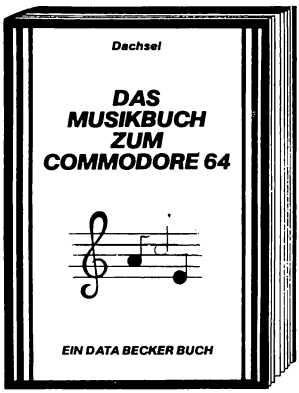
64 Tips & Tricks ist eine hochinteressante Sammlung von Anregungen zur fortgeschrittenen Programmierung des COMMODORE 64, Poke's und anderen nützlichen Routinen, interessanten Programmen sowie aktuellen Programmiertips und -tricks. Aus dem Inhalt: 3D-Graphik in BASIC — Farbige Balkengraphik — Definition eines eigenen Zeichensatzes — Die Tastaturbelegung und ihre Änderung — Dateneingabe mit Komfort — Simulation der Maus mit einem Joystick — BASIC für Fortgeschrittene — Ihr COMMODORE 64 spricht Deutsch — CP/M auf dem COMMODORE 64 — Druckeranschluß über den USER-Port — Datenübertragung von und zu anderen Rechnern — Der Expansionsport — Synthesizer in Stereo — Retten einer nicht ordnungsgemäß geschlossenen Datei — Erzeugen einer BASIC-Zeile in BASIC — Der Kassettenpuffer als Datenspeicher — Sortieren von Stringfeldern — Multitasking auf dem COMMODORE 64 — POKE's und die Zeropage — GOTO, GOSUB und RESTORE mit berechneten Zeilennummern INSTR und STRING-Funktion — Repeatfunktion für alle Tasten — und vieles andere mehr. Alle Maschinenprogramme mit BASIC-Ladeprogrammen. 64 TIPS & TRICKS ist eine echte Fundgrube für jeden COMMODORE 64 Anwender.

64 INTERN,
3. Auflage,
1983, ca. 320 Seiten,
DM 69,-

64 INTERN ist unser großes Buch zum COMMODORE 64. Ideal für alle, die sich näher mit Programmierung, Technik und Betriebssystem von COMMODORE's Supermaschine auseinandersetzen wollen. Detailliert werden Architektur und technische Möglichkeiten des C-64 beschrieben, vom Prozessor 6510 über Video-Controller 6569, Sound Controller 6581, Ein-/Ausgabesteuerung mit CIA 6526 bis hin zum seriellen IEC-Bus, USER-Port und EXPANSION-PORT. Paralell dazu wird die Programmierung der hochauflösenden Graphik und der Sprites erklärt, die Sonderprogrammierung und die Programmierung des C-64 in Maschinensprache. Mit einen ausführlich dokumentierten ROM-Listing wird das Betriebssystem zerlegt. Wertvolle Hinweise zum Umsetzen von Programmen bringt ein Vergleich zwischen VC-20, C-64 und CBM 8000. Zahlreiche lauffertige Beispielprogramme und Routinen, z. B. Graphik-Aid, HARDCOPY, RENEN und PRINT USING runden das Buch ab. Der Clou von 64 INTERN sind neben den zahlreichen Schaltbildern und Blockdiagrammen 2 Original COMMODORE-Schaltpläne zum Ausklappen, die zusätzlich sehr ausführlich beschrieben und dokumentiert sind. Dieses Buch sollte jeder COMMODORE 64- Anwender und Interessent haben.

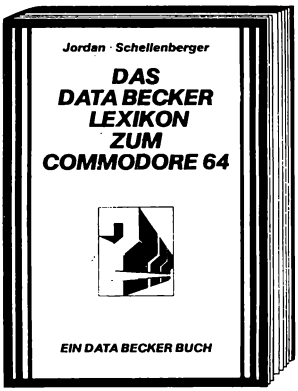
MIT DATA BECKER BÜCHERN MACHEN

SIE MEHR AUS IHREM COMMODORE



Der COMMODORE 64 ist ein Musikgenie. Mit diesem Buch lernen Sie alles über seine musikalischen Fähigkeiten und wie Sie diese programmtechnisch nutzen. Der Inhalt des Musikbuches reicht von einer Einführung in die Computermusik über die Erklärung der Hardware-Grundlagen und die Programmierung in BASIC bis hin zur fortgeschrittenen Musikprogrammierung in Maschinensprache. Zahlreiche Beispielprogramme, komplette Songs und nützliche Routinen ergänzen den leicht verständlichen Text. Geschrieben wurde das Buch von Thomas Dachsel, dem Autor der weltbekannten Musikprogramme SYNTHIMAT und SYNTHESOUND. Erschließen Sie sich die Welt des Sounds und der Computermusik mit dem Musikbuch zum COMMODORE 64.

DAS MUSIKBUCH ZUM
COMMODORE 64
1984, ca. 200 Seiten, DM 39,—



So etwas haben Sie gesucht: Umfassendes Nachschlagewerk zum COMMODORE 64 und seiner Programmierung. Allgemeines Computerlexikon mit Fachwissen von A-Z und Fachwörterbuch mit Übersetzungen wichtiger englischer Fachbegriffe - das DATA BECKER LEXIKON ZUM COMMODORE 64 stellt praktisch drei Bücher in einem dar. Es enthält eine unglaubliche Vielfalt an Informationen und dient so zugleich als kompetentes Nachschlagewerk und als unentbehrliches Arbeitsmittel. Viele Abbildungen und Beispiele ergänzen den Text. Ein Muß für jeden COMMODORE 64 Anwender.

DAS DATA BECKER
LEXIKON ZUM
COMMODORE 64
1984, ca. 350 Seiten, DM 49,—

Graphik ist eine der Hauptstärken des COMMODORE 64. Mit diesem leicht verständlich geschriebenen, aber sehr umfassenden neuen Buch lernen Sie, wie Sie die graphischen Fähigkeiten für Ihre eigenen Programme optimal nutzen. Der Inhalt reicht von den Grundlagen der Graphikprogrammierung über das Erzeugen einfacher Figuren, die Arbeit mit Sprites, Zeichensatzprogrammierung, Hardcopy und IRQ-Handhabung bis hin zur Funktionendarstellung, Laufschrift, Statistik, 3-D, CAD, den Geheimnissen der Actionsspiele und Lightpenanwendungen. Zahlreiche Beispielprogramme, nützliche Routinen und komplette Anwendungen ergänzen diese Buch, das die faszinierende Welt der Computergraphik jedermann zugänglich macht. Geschrieben wurde das Graphikbuch von dem bekannten SUPERGRAPHIK-Autor Axel Plenge.

DAS GRAPHIKBUCH
ZUM COMMODORE 64 1984, ca. 250 Seiten, DM 39,—

Das sollte Ihr erstes Buch zum COMMODORE 64 sein: 64 FÜR EINSTEIGER ist eine sehr leicht verständliche Einführung in Handhabung, Einsatz, Ausbaumöglichkeiten und Programmierung des COMMODORE 64, die keinerlei Vorkenntnisse voraussetzt. Sie reicht vom Anschluß des Geräts über die Erklärung der einzelnen Tasten und Funktionen sowie die Peripheriegeräte und ihre Bedienung bis zum ersten Befehl. Schritt für Schritt führt das Buch Sie in die Programmiersprache BASIC ein, wobei Sie nach und nach eine komplette Adressenverwaltung erstellen, die Sie anschließend nutzen können. Zahlreiche Abbildungen und Bildschirmfotos ergänzen den Text. Viele Anwendungsbeispiele geben nützliche Anregungen zum sinnvollen Einsatz des COMMODORE 64. Das Buch ist sowohl als Einführung als auch als Orientierung vor dem 64er Kauf gut geeignet.

64 FÜR EINSTEIGER
1984, ca. 200 Seiten, DM 29,—

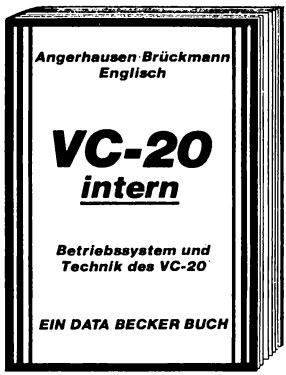
Diese neue, umfangreiche Programmsammlung hat es in sich. Über 50 Spitzenprogramme für den COMMODORE 64 aus den unterschiedlichsten Bereichen, von attraktiven Superspielen („Senso", „Pengo", „Seeschlacht", „Poison Square", „Memory") über Graphik- und Soundprogramme („Fourier 64", „Akustograph", „Funktionsplotter") sowie Utilities („SORT", „Renumber", „Disk Init", „Menue") bis hin zu kompletten Anwendungsprogrammen wie „Videothek", „File Manager" und einer komfortablen Haushaltsbuchführung, in der fast professionell gebucht wird. Der Hit sind zu jedem Programm aktuelle Programmiertips und Tricks der einzelnen Autoren zum Selbermachen. Also — nicht nur abtippen, sondern auch dabei lernen und wichtige Anregungen für die eigene Programmierung sammeln.

DATA BECKER's GROSSE 64er PROGRAMMSAMMLUNG
1984, ca. 250 Seiten, DM 49,—

Achtung Hobbyelektroniker: Dieses neue Buch enthält nicht nur alles über die Ausbaumöglichkeiten des COMMODORE 64 und wie man ihn über seine Schnittstellen mit dem „Rest der Welt" verbindet, sondern auch umfassende Informationen über die vielfältigen Einsatzmöglichkeiten des COMMODORE 64 von der Lichtorgel über Motorsteuerung, Spannungs- und Temperaturmessung bis zur programmierbaren Stromversorgung, und wie man diese verwirklicht. Umfassende Darstellung der Grundlagen. Als Clou enthält das Buch zehn komplette Schaltungen zum Selberbauen, vom Eprommer über Eprom-Karte, Logic Analyzer, Frequenzzähler, Hardware-Tracer, Pulsmeßgerät, Klatschschalter und Digital-Voltmeter bis zur preiswerten Spracheingabe-Sprachausgabe. Jeweils komplett mit Schaltplan, Layout und Softwarelisting. Auch für Profis interessant.

DER COMMODORE 64 UND
DER REST DER WELT 1984, ca. 220 Seiten, DM 49,—

DATA BECKER BÜCHE



VC-20 INTERN,
2. Auflage,
1983, ca. 230 Seiten,
DM 49,-

Die bereits 3. Auflage von VC-20 INTERN ist wieder erheblich erweitert worden. Das Buch beschäftigt sich ausführlich mit der Technik und dem Betriebssystem des VC-20. Dazu gehört natürlich zuerst einmal ein komplettes und ausführlich dokumentiertes ROM-Listing. Dazu gehört auch die Belegung der ZEROPAGE, dem wichtigsten Speicherbereich für den 6502-Prozessor, eine übersichtliche Auflistung der Adressen aller Betriebssystemroutinen, ihrer Bedeutung und ihrer Übergabeparameter. Dies ermöglicht dem Programmierer endlich, den VC-20 von Maschinensprache aus sinnvoll einzusetzen. Denn warum Routinen, die bereits vorhanden sind, noch einmal schreiben? Für Einsteiger befindet sich eine Einführung in die Programmierung in Maschinensprache am Anfang des Buches. Diese behandelt die Themen Maschinensprachemonitor, Disassembler und Assembler. Außerdem wird die Verbindung von BASIC- und Maschinenspracheprogrammen besprochen. Doch nicht nur die Software, auch die Hardware wird ausführlich beschrieben. Detailliert werden alle wichtigen IC's im VC-20 in Ihrer Arbeitsweise beschrieben. Neben einem übersichtlichen Blockschaltbild enthält VC-20 INTERN noch als besonderen Clou drei Original COMMODORE Schaltpläne zum Ausklappen. Dieses Buch braucht jeder, der sich näher mit Technik und Maschinenprogrammierung des VC-20 auseinandersetzen möchte.

VC-20 TIPS & TRICKS,
2. Auflage,
1983, ca. 230 Seiten,
DM 49,-

Die überarbeitete und erheblich erweiterte 2. Auflage von VC-20 TIPS & TRICKS enthält endlich eine genaue Erklärung des Speicheraufbaus des VC-20 in allen möglichen Erweiterungsstufen. Vergessen Sie alles, was Ihnen über Grafik und Sound auf dem VC-20 bisher geliefert wurde. In diesem Buch finden Sie von den Grundlagen angefangen über die programmierbaren Zeichen, über eine Grafikerweiterung in Maschinensprache, über luxuriöse Programme wie z. B. Grafikeditor, Funktionenplotter oder 3D-Grafiken bis hin zur Supererweiterung zum VC-20 alles Wissenswerte und zum Arbeiten Notwendige für eine gute Grafik. Ebenso ausführlich wird die Tonerzeugung behandelt. Neben der Erläuterung der Grundlagen und einem Soundeditor sind natürlich auch fertige Programme, z. B. der VC-20 als Synthesizer oder Schlagzeug abgedruckt. Auf über 100 Seiten finden Sie viele nützliche Tricks, POKE-Befehle, BASIC-Erweiterungen fix und fertig zum Eintippen. Dazu gehören z. B. Unnew, automatische Zeilennumerierung, die Programmierung des User-Port, ein Diskmenue, Hardcopy und so weiter. Bringen Sie Ihrem VC-20 völlig neue Befehle bei! Abgerundet wird das Buch durch Anwenderprogramme zum Eintippen, wie Dateiverwaltung, Textverarbeitung und Spiele. VC-20 TIPS & TRICKS braucht jeder VC-20 Besitzer, der mehr über seinen Computer wissen möchte.

ACHTUNG AUTOREN

SIE haben ein gutes Programm oder ein Manuskript zu einem interessanten Buch geschrieben oder würden dies gerne tun

SIE würden das Ergebnis Ihrer Arbeit gerne in größeren Stückzahlen vermarktet sehen

SIE suchen dafür den leistungsfähigen Verlag und Vertriebspartner Ihres Vertrauens

WIR besitzen große Erfahrung in der professionellen Vermarktung von Software und Literatur, nicht nur in Deutschland, sondern auch weltweit

WIR haben in den letzten 12 Monaten mit dem Verkauf von 200.000 Büchern und 50.000 Programmen unsere Leistungsfähigkeit unter Beweis gestellt

WIR suchen weitere Autoren, mit denen wir gemeinsame Erfolge erringen können

SIE und **WIR** sollten zusammenarbeiten.

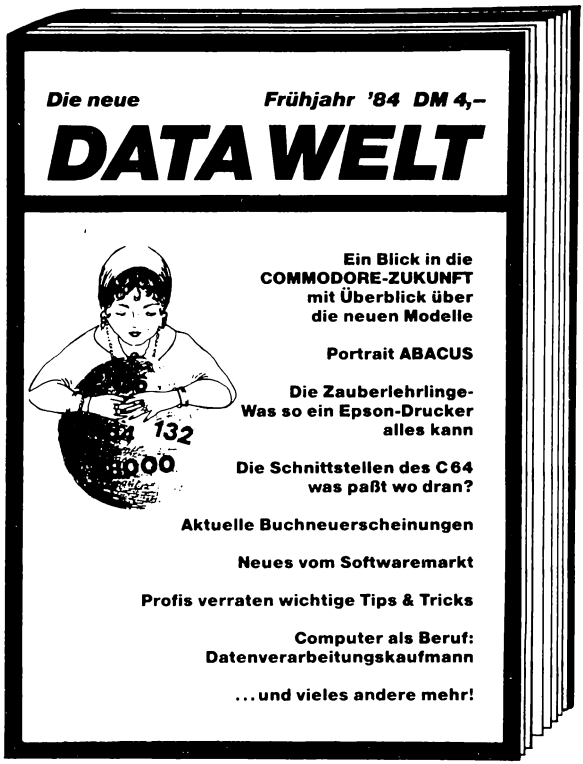
Bitte, senden Sie eine Beschreibung Ihres Programms oder Ihres Buchprojekts an Dr. Achim Becker c/o DATA BECKER, oder fordern Sie einfach unsere unverbindlichen „Informationen für Autoren" an.

IHR GROSSER PARTNER FÜR KLEINE COMPUTER

DATA BECKER

Merowingerstraße 30 · 4000 Düsseldorf · Telefon (02 11) 31 00 10 · im Hause AUTO BECKER

Darauf haben Sie gewartet!



Die neue DATA WELT ist eine Computerzeitschrift speziell für COMMODORE-Anwender. Brandaktuell (detaillierte Informationen über die neuen COMMODORE Computer 264 und 364) und randvoll mit Berichten, Trends und interessanten Programmiertips. 80 Seiten stark im Magazinformat. Gleichzeitig als Nachfolger des VC-Infos umfassende Übersicht über aktuelle Produkte, Bücher und Programme rund um COMMODORE 64 und VC-20. Die Frühjahrsausgabe der neuen DATA WELT erhalten Sie ab Anfang März überall dort, wo es DATA BECKER BÜCHER und -PROGRAMME gibt. Am besten gleich holen oder direkt bei DATA BECKER gegen DM 4,– in Briefmarken anfordern.
Weitere Erscheinungs-Termine: Ausgabe Sommer – Ende Juni 84 · Ausgabe Herbst – September 84 · Ausgabe Winter – Ende November 84.

This was brought to you

from the archives of

http://retro-commodore.eu